MAKE | BUILD | HACK | CREATE

HackSpace

TECHNOLOGY IN YOUR HANDS

hsmag.cc | March 2022 | Issue #52

PCB DESIGN

Create custom footprints for form and function

MORSE CODE

40

RASPBERRY PI PICO PROJECTS

How to get the most out of your microcontroller

LIGHT! CAMERA! ROBOT!

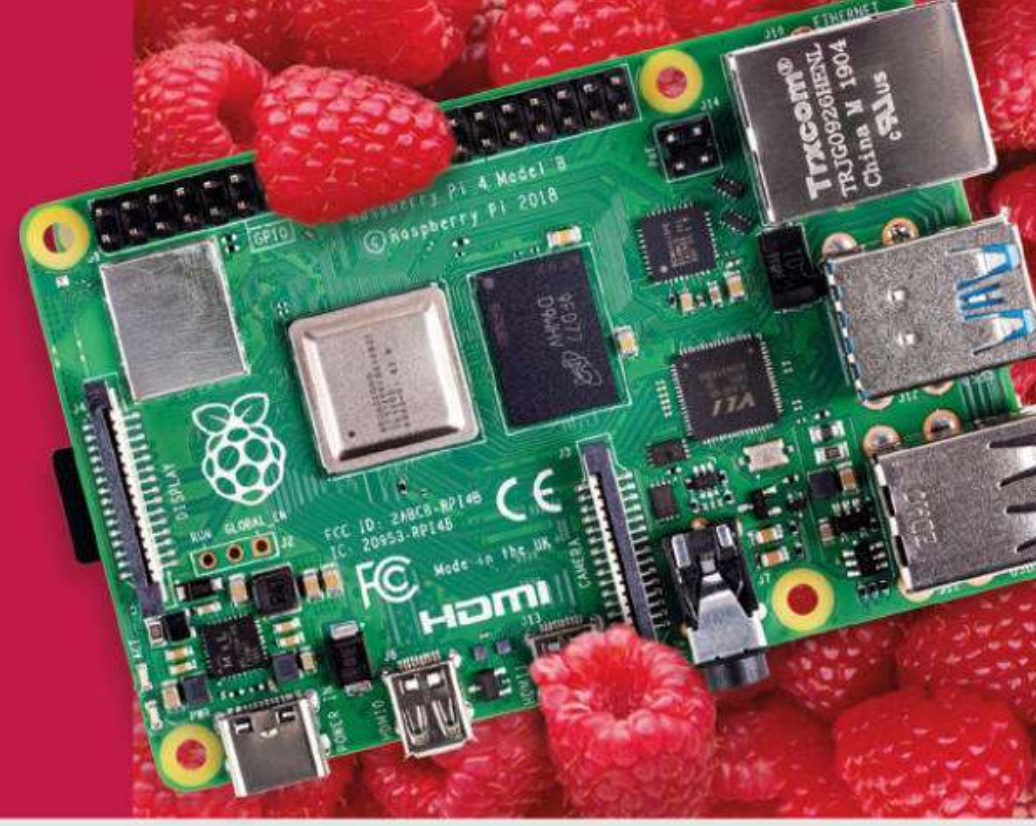
An engineering musical extravaganza

LASER CUTTING

zap zap, cut, cut

LEDS PGA2040 SENSORS FILAMENT

American **Raspberry Pi** Shop



- Displays
- HATs
- Sensors
- Cases
- Arcade
- Swag
- Project Kits
- Cameras
- Power Options
- Add-on Boards
- Cables and Connectors
- GPIO and Prototyping

Partner and official reseller for top Pi brands:

adafruit · sparkfun ELECTRONICS · PIMORONI · HiFiBerry · and many others!

Price, service, design, and logistics support for **VOLUME PROJECTS**

USA **PiShop.us**

Canada **PiShop.ca**

Raspberry Pi APPROVED RESELLER

Welcome to HackSpace magazine

It's been a year since Raspberry Pi Pico launched on the cover of this magazine, and it's fair to say that the impact on the hobbyist community has been huge. It's a cheap board with a powerful set of I/O options that's easy to work with in a wide range of programming environments.

As the RP2040 – the microcontroller at the heart of Pico – was designed from the ground up in-house, it's got some unique features that make it particularly powerful, including the Programmable I/O interface. Now, a year on, we're seeing this hardware being pushed to its full capabilities as makers have had time to understand it and get used to its features.

It's got some unique features that make it particularly powerful, **including the Programmable I/O interface**

In this issue, we're taking a look at some of the best projects built on this new microcontroller. Read on for some inspiration and ideas for what you could work on throughout the year.

BEN EVERARD
Editor ben.everard@raspberrypi.com

Got a comment, question, or thought about HackSpace magazine?

get in touch at **hsmag.cc/hello**

GET IN TOUCH

hackspace@raspberrypi.com
hackspacemag
hackspacemag

ONLINE

hsmag.cc

Available on the App Store

GET IT ON Google Play

PAGE 42

EDITORIAL

Editor
Ben Everard
ben.everard@raspberrypi.com

Features Editor
Andrew Gregory
andrew.gregory@raspberrypi.com

Sub-Editors
David Higgs, Nicola King

DESIGN

Critical Media
criticalmedia.co.uk

Head of Design
Lee Allen

Designers
Sam Ribbits, Olivia Mitchell, Ty Logan

Photography
Brian O'Halloran

CONTRIBUTORS

Jo Hinchliffe, Marc de Vinck, Rob Miles, Andrew Lewis, Rosie Hattersley, Phil King, Mike Bedford

PUBLISHING

Publishing Director
Russell Barnes
russell@raspberrypi.com

Advertising
Charlie Milligan
charlotte.milligan@raspberrypi.com

DISTRIBUTION

Seymour Distribution Ltd
2 East Poultry Ave,
London EC1A 9PT
+44 (0)207 429 4000

SUBSCRIPTIONS

Unit 6, The Enterprise Centre,
Kelvin Lane, Manor Royal,
Crawley, West Sussex, RH10 9PE

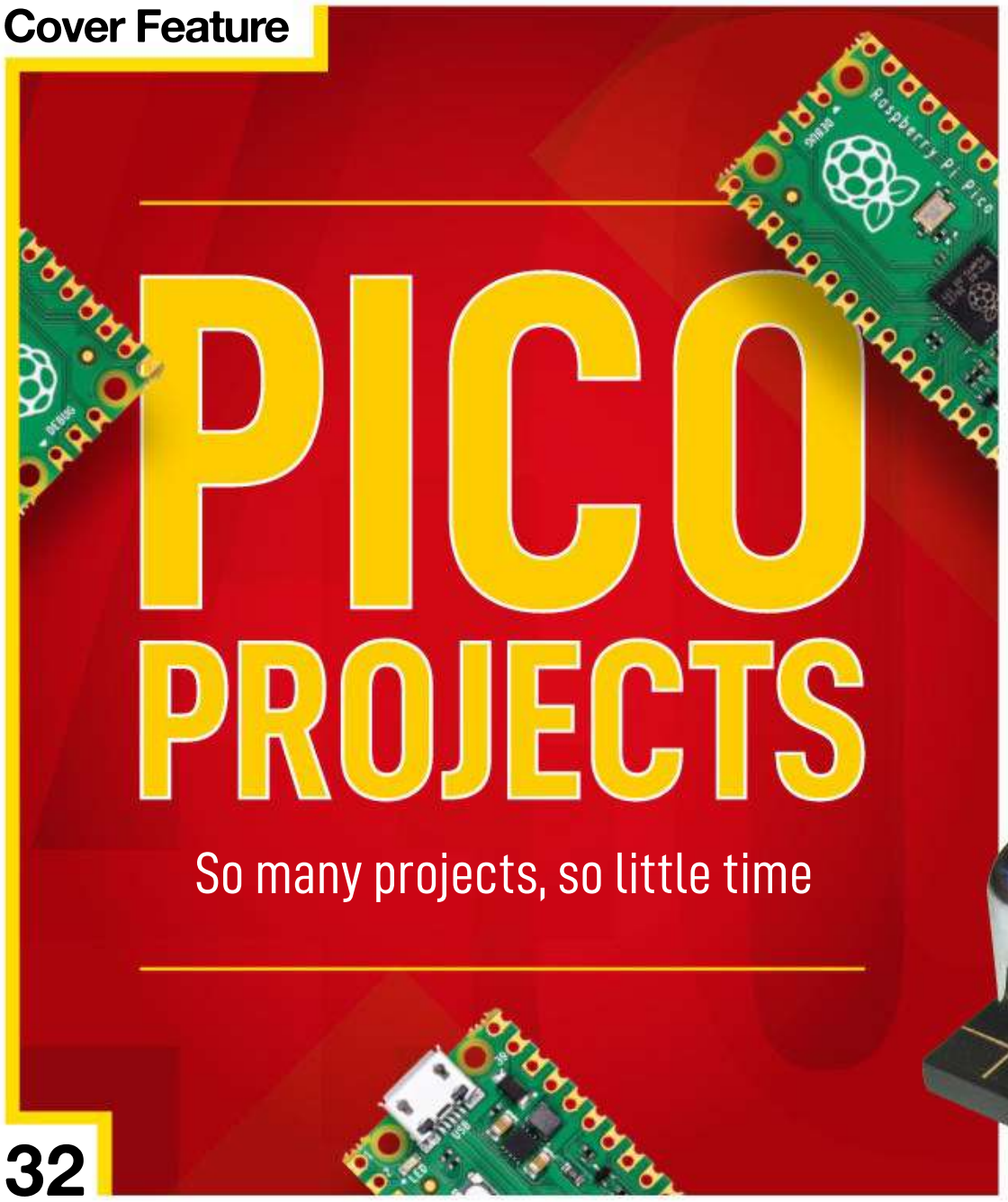
To subscribe
01293 312189
hsmag.cc/subscribe

Subscription queries
hackspace@subscriptionhelpline.co.uk

Contents

62

Cover Feature



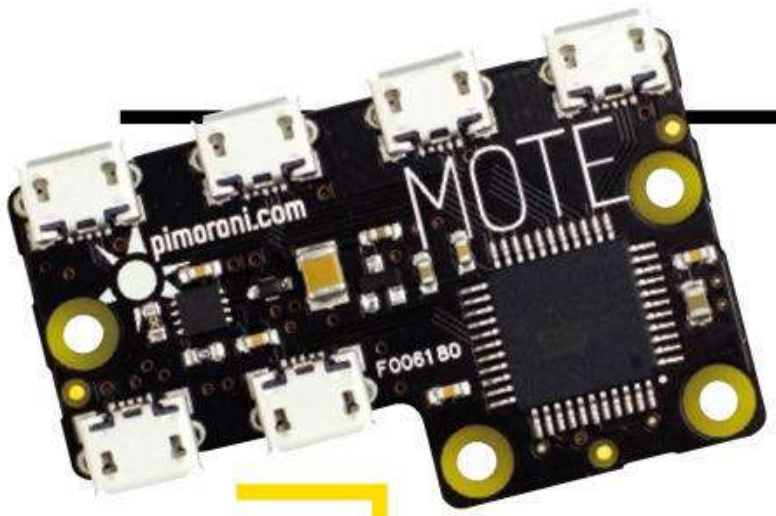
32

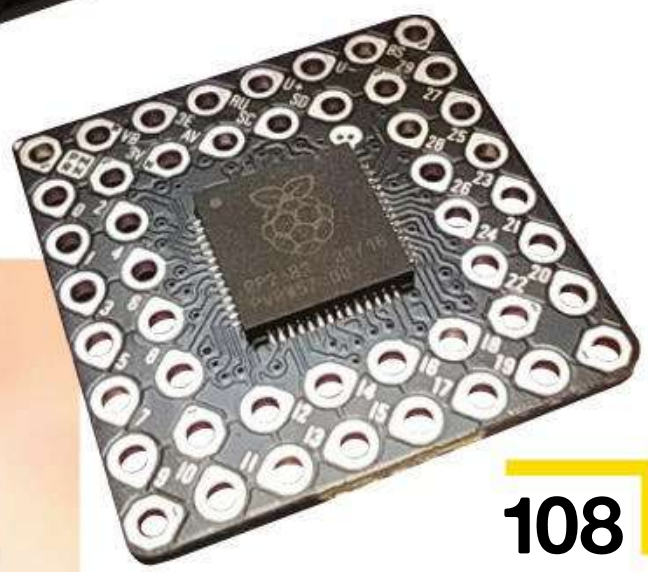
Tutorial

Animated GIFs

70 Pro tip: pronounce it 'jif' to annoy everyone in earshot

84





Review

QT Py ESP32-S2

Interview

Odd Jayy

67 FORGE

101 FIELD TEST

REGULAR

Happy birds

By Papy et les Resistances

hsmag.cc/IOTBirdFeeder

There's a lot to take in with this build, so we'll go slowly. It's 3D-printed. It's solar-powered. It's connected to the Internet of Things. It takes photographs of birds when they land on the bird feeder, and then sends those photographs to your smartphone via a free app. Most of all, it looks great: it's not a proof of concept or a slightly wobbly prototype.

The two solar panels on the roof charge a 4800mAh Li-ion battery, which powers an ESP32-CAM board. This sleeps in low-power mode until a PIR sensor detects a bird, whereupon it wakes up and takes a picture.

Right
In Papy's Paris garden, the power from two 90mm × 60mm solar panels supplying 100mA each is enough to make this project work; in less sunny climes, he recommends two panels of 135mm × 165mm supplying 583mA each

REGULAR

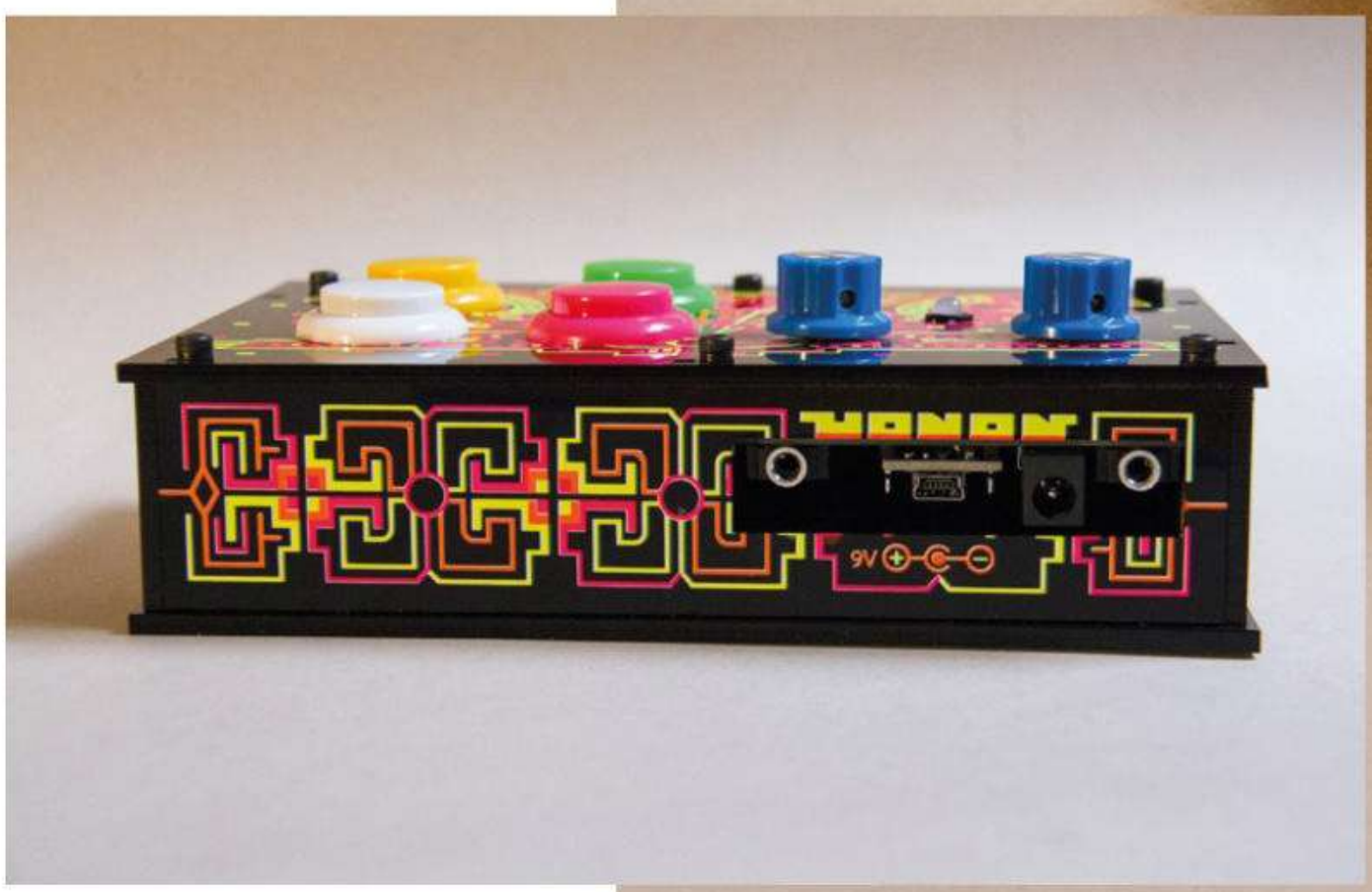
Hanan Cumbia

By Oficina de Sonido

hsmag.cc/HananCumbia

Cumbia is a quite brilliant genre of music originally from Colombia, and now popular throughout Latin America. This drum machine by Oficina de Sonido, with graphic design by Yeffry Ruta Mare and screen printing by Los Laberintos, is the sound of cumbia in a box – an open-source hardware box.

It has just four sounds: kick, güiro, cow-bell, and conga, with pitch control for cow-bell and conga. Great big click buttons remind you that you're playing a musical instrument, not programming a machine. More than anything, we love the effort that has clearly gone into this build.



Right
Seriously, go and listen to some cumbia now; it's great!


CUMBIA

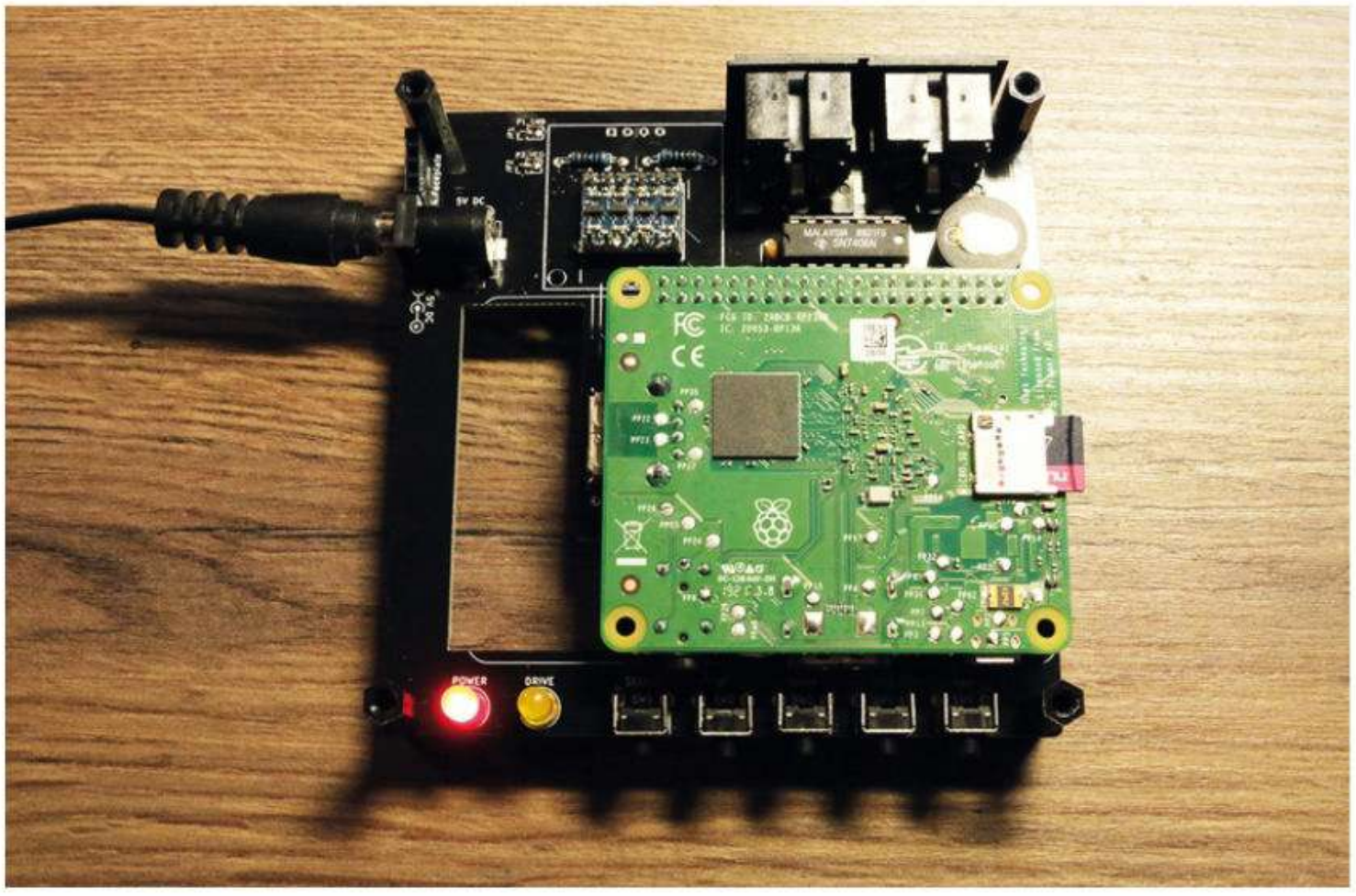
REGULAR

Commodore C64 disk drive emulator

By RetroFletch

hsmag.cc/C64DiskDriveEmulator

Every obsolete computing platform that there has ever been has a bunch of hardcore enthusiasts, for whom the zeroes and ones of their youth are somehow better than the crude, modern equivalents. We've seen loads of Spectrum emulators/re-creations, so it's only fair that we should show off this piece of hardware that pays tribute to that other titan of 1980s computing, the Commodore C64. It's a disk drive emulator that enables the user in 2022 to load data into their treasured C64 (or C128) without the need for a clanking mechanical unit. For a certain niche it's going to be very useful indeed, as the one thing you can guarantee about magnetic disks is that they will degrade over time.



Right
This drive uses a Raspberry Pi Model 3A+ or 3B+


POWER
DRIVE
SELECT
UP
DOWN
EXIT
ADD


PI 1541-II
DISK DRIVE
DS/DD
Soft Sectored
40 Tracks per side
POWER
DRIVE
SELECT
UP
DOWN
EXIT
ADD

REGULAR

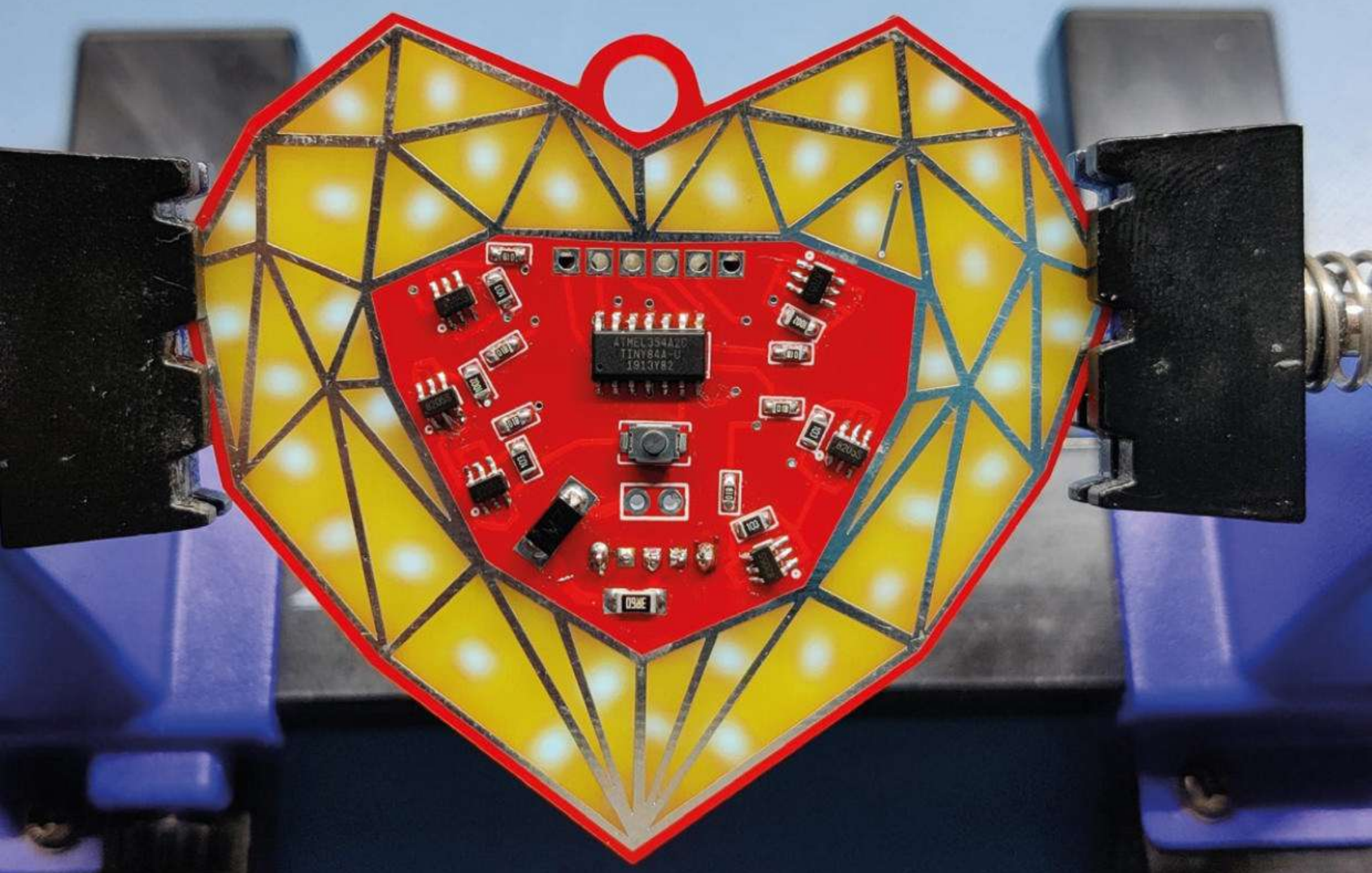
PCB heart pendant

By Arnov Sharma

hsmag.cc/PCB-Heart

We've been trying our hand at translucent PCBs in recent weeks – see HackSpace issue 51 for more – and so has Arnov Sharma, whose PCB heart pendant uses 36 surface-mounted LEDs controlled by an ATtiny84 MCU. The LEDs are wired in series in six groups of six, so they can't be controlled individually, but can be programmed to flash in the zones of the PCB they occupy.

Right
Arnov designed the custom PCB in Fusion 360


ATMEL354A2D
TINY84A-U
1913Y82

“We wanted to show that there’s more to engineering than men in brown overalls”

Take a bunch of five-year-olds, two engineers and a dash of the old razzle-dazzle, and what do you get? Kids Invent Stuff’s toilet-cleaning robot and a musical extravaganza!

You may be familiar with the work of Ruth Amos and Shawn Brown; they’re the dynamic duo behind the brilliant Kids Invent Stuff YouTube channel. The premise is simple enough: every month they invite kids to send them invention ideas on a particular theme; at the time of writing, they’d just finished accepting entries for a winter-themed challenge. Ruth and Shawn (who are both real, adult engineers with real jobs) then turn the winning idea into a real product. Past creations have included a wearable air bed, a motorised magic carpet, and a saxophone that fires glitter. Check it out for yourself at **kidsinventstuff.com**.

One day, the pair decided to do something different. They’d take the original idea that works so well, and add a musical element.

“The whole point of what we do is to show kids that they can be engineers,” the pair told us. “We’ve not been going long enough to have inspired a six-year-old who’s now studying engineering at university, but we have had lots of feedback from teachers and parents, telling us how kids have been inspired to try more things. And it’s not just that – it’s great for kids’ confidence too, coming up with ideas and having them taken seriously by two adults who build them as they describe them to us.”

Kids Invent Stuff has been going for a while, but recently, they had a go at something different: a musical. “We both love musical →

Right
The robot works via remote control

REGULAR

theatre, and it seemed about as far away as you can get from the stereotype of what engineers do. That made it seem like a good way of reaching people we wouldn't otherwise be talking to. We took our inspiration from *Horrible Histories*, or the *Hamilton* musical – yes, it's fun, but it's also a subtle way to get information over to a new audience about what an engineer does."

They kept the structure the same – the idea for a toilet-cleaning robot came from Erin, a first-year reception pupil at a school in Cornwall, which Ruth and Shawn built in segments and assembled for the filming. The builds are usually a logistical challenge at the best of times, but for this build, the pair had to add singing, dancing, music, lyrics, and the dangerous element of smashed toilet seats.

"We recorded the questions with the kids, then sent the kids' answers over to Seamas Carey (**seamascareymusic.com**), a musician we know. It was a lot more work to incorporate the extra elements – someone had to film and direct it, someone else had to write the music, sing, and dance – but it was also refreshing to share the creative process with other people.

That's the musical taken care of; how does the robot perform?

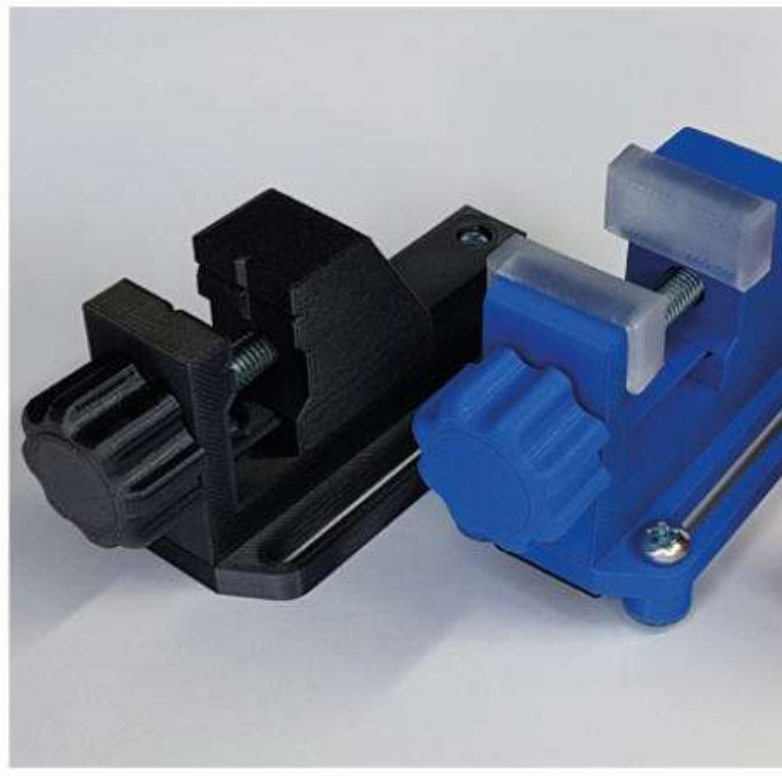
"It does work; it's pretty brutal actually. Your toilet would be clean, but it would have the enamel worn off it pretty swiftly".

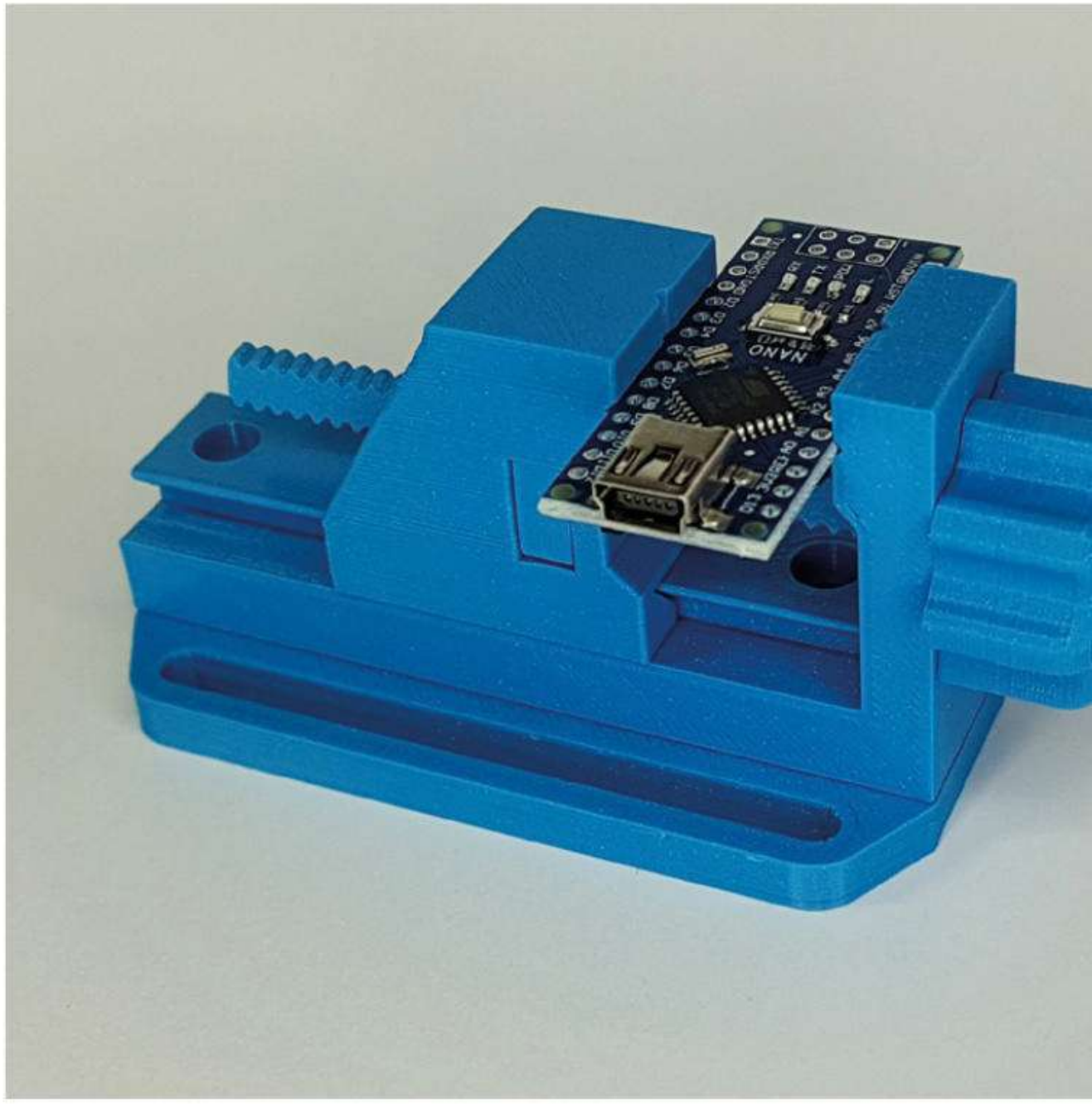
Watch the whole thing, and admire Shawn's dog, Luna, here: **hsmag.cc/RobotMusical**.

Below
At the time of writing, Erin hadn't seen her robot in action – apparently kids like to know that the builds are real, because they're accustomed to so much CGI on TV

Objet 3d’art

3D-printed artwork to bring more beauty into your life

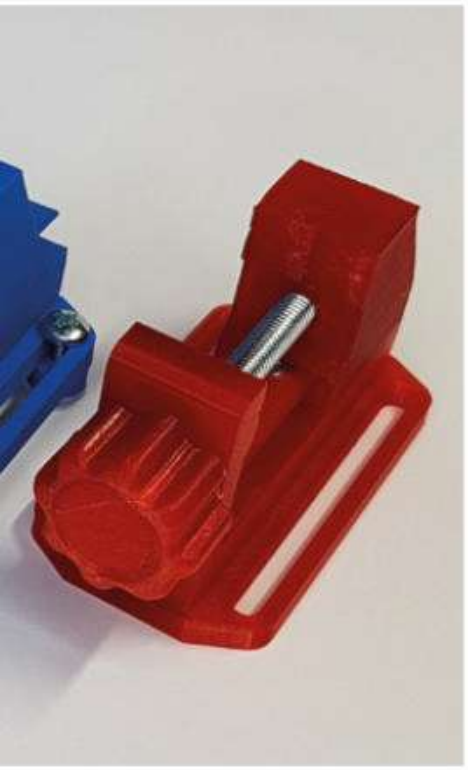


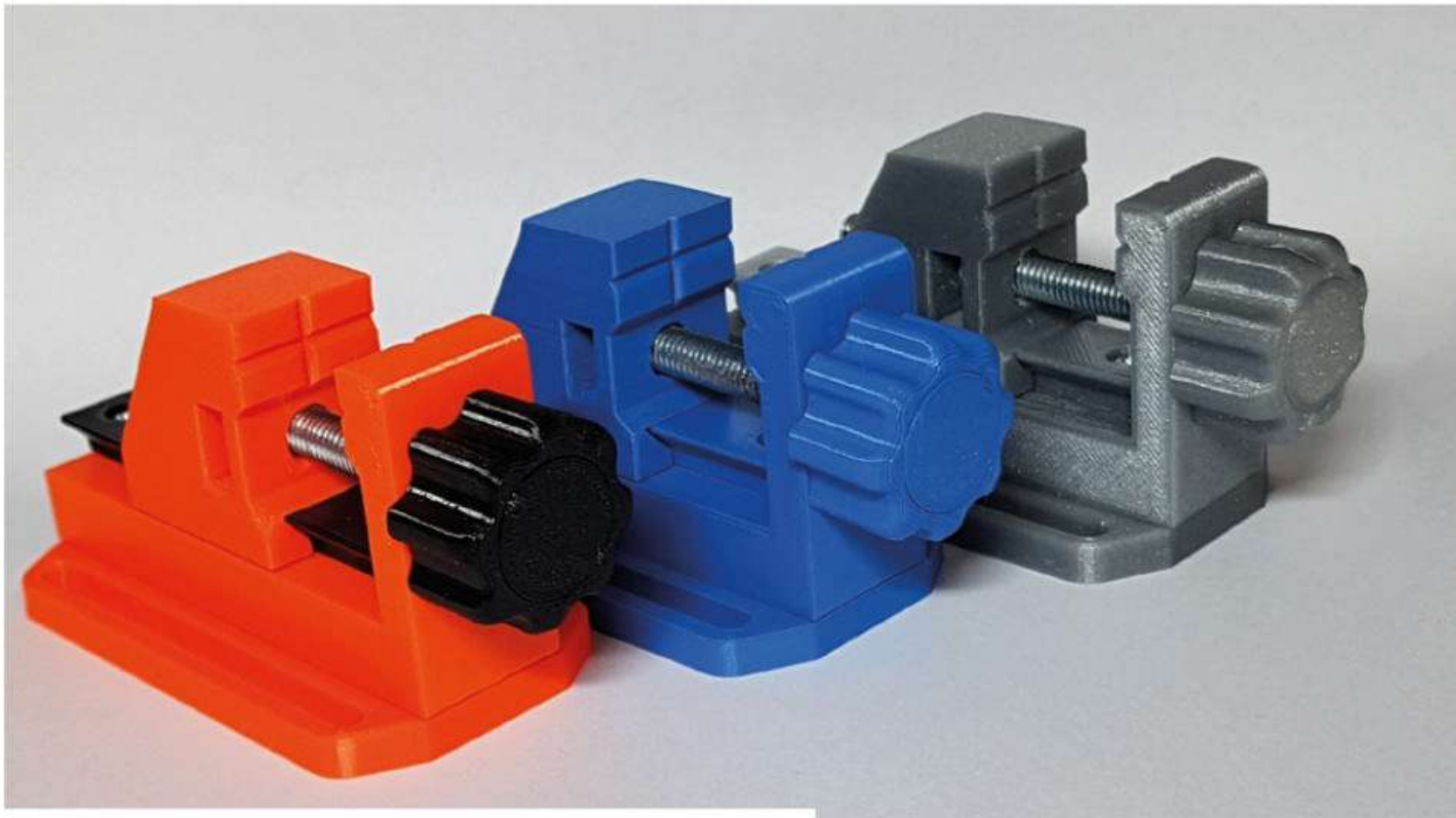


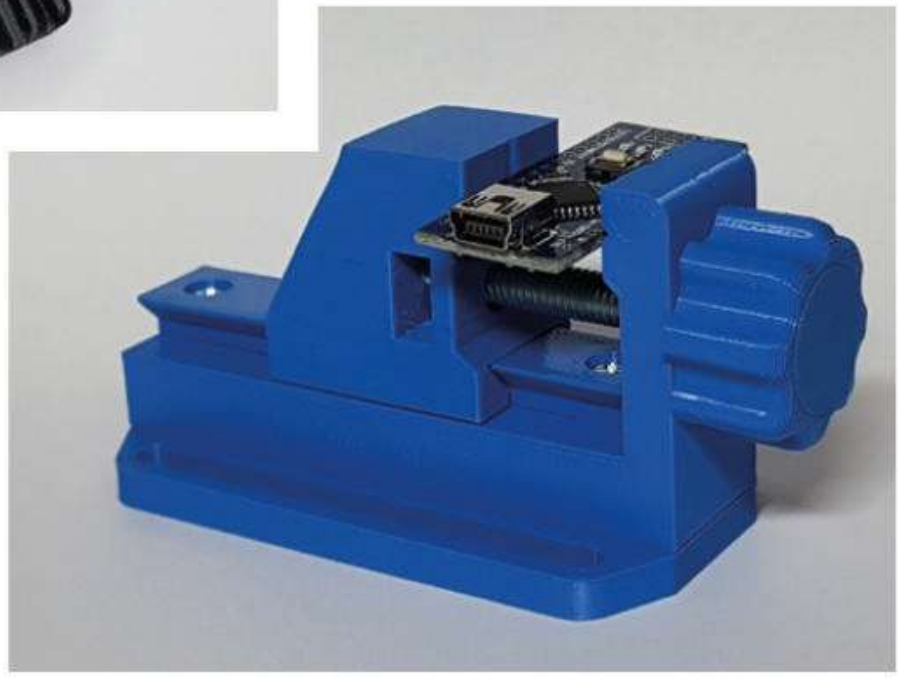
This clamp by HD_Creator is extremely simple, which is why we rate it so highly (that, and the coincidence that we need a couple of clamps right now). It’s a refinement of an earlier work by the same maker, this time with a smoother action, a larger capacity jaw, and a version comprised of 100% 3D-printed parts, rather than the mix of 3D-printed and metal parts shown in most of these images.

If you like the look of this and decide to add metal parts, you’ll need an M6 × 70 bolt and an M6 nut, plus three M3 × 16 screws to attach the clamp to your work surface. PETG is recommended; as you’d expect for a build that’s meant to carry some force through it, it’s printed at 100% infill.

hsmag.cc/MiniatureVice







REGULAR

Meet The Maker: Carl Bugeja

Testing flexible PCBs to their limits

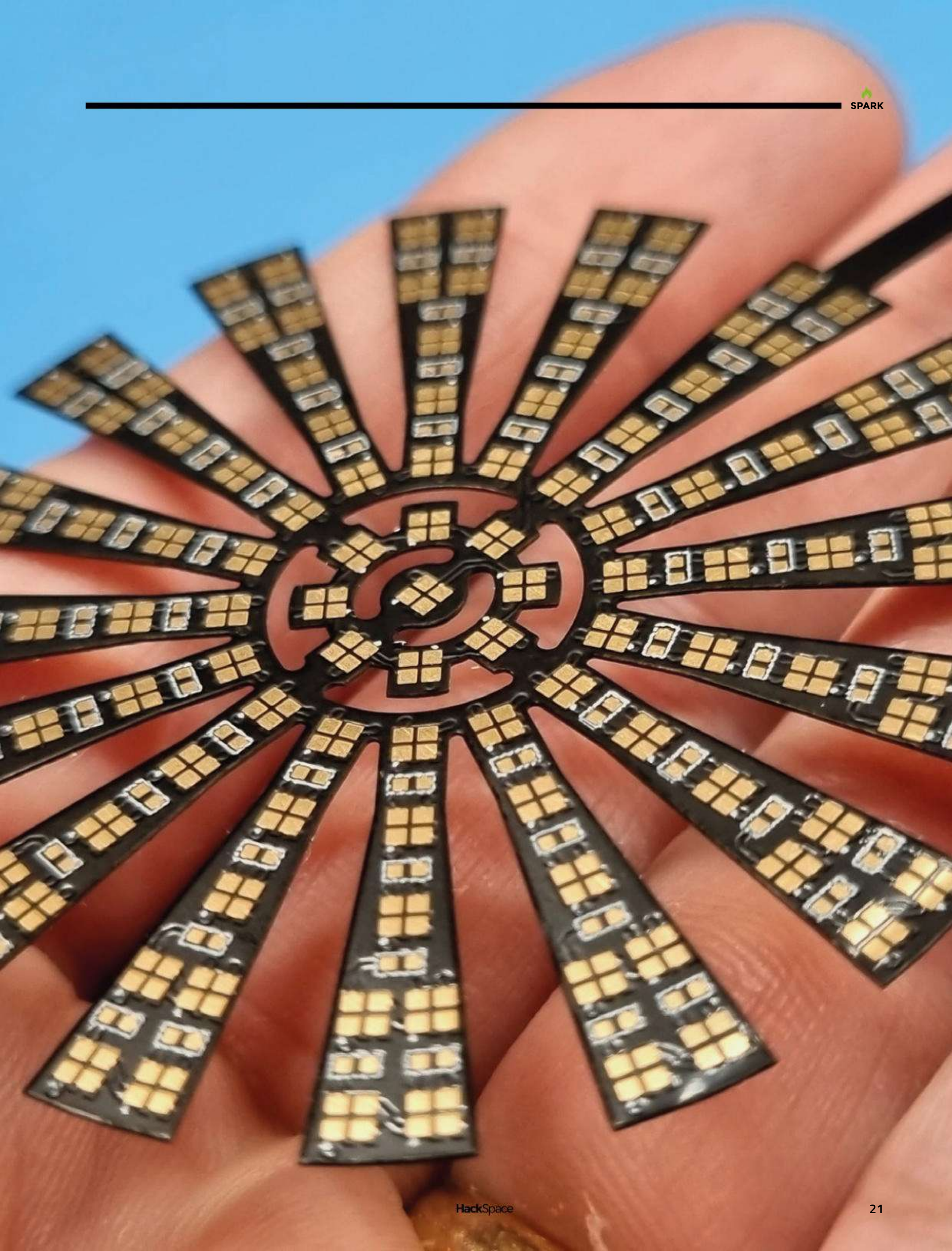
We're used to seeing electronics as something that gets added on to a machine to make it work – think of the robot that's just a pile of servos until you drop a microprocessor and some wires into it to make it function. But there's another way: instead of building the device, then adding the functionality, Carl Bugeja is an electronics engineer making robots out of PCBs. Two-layer PCBs, twelve-layer PCBs, flexible PCBs – he's doing fascinating work, messing about with things just because he can, and recording the results. His work is open-source hardware, meaning anyone can copy what he does, but we're just happy to keep an eye on his YouTube channel (**hsmag.cc/CarlBugeja**) and see what he comes up with next. We caught up with Carl to find out how things are going, what he's currently working on, and what we can expect to see from him next. Here's what he said:

> What if we could embed the coils with the PCB tracks themselves? **That's how it got started**

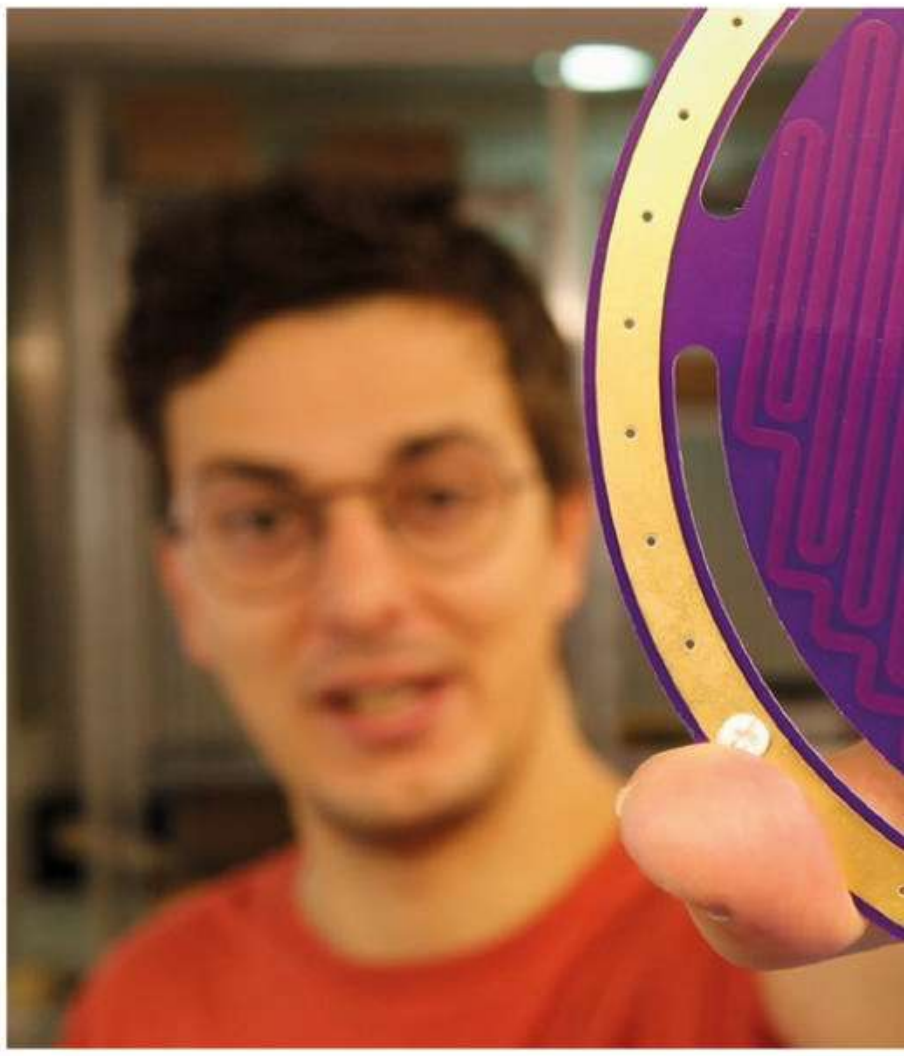
"I graduated in 2016, and right after that, I was working on a startup; we were mainly building really small drones. So the main challenge for this project was miniaturisation and simplifying things to the utmost. The startup ended a few months after we started because of financial reasons. But by then, simplifying electronics and products and stuff like that had become my passion.

"So I kept thinking about the drone and how it could be simplified. And that's when I came up with the idea of the PCB motor. I didn't know it at the time, but PCB motors were already available – I only know that because after I published my project, people started sending me some other PCB motors that were commercially available. But when I made my first version, all I knew was that there was a motor that had some actuators that could be connected to a PCB.

"So that was my first idea. That's when I first heard about PCB motors. But then I said, what if we could embed the coils with the PCB tracks



REGULAR



Right
Carl's also attempting to make a PCB heater for reflowing solder

themselves? That's how it got started. The first version worked at the first attempt – that doesn't usually happen with electronics!

"The main challenges are the thermal issues when it comes to PCB motors because the tracks tend to get hot – this happens in other motors as well. But obviously, when you have a PCB, there's a limit to how much you can heat it. I think that's the main limitation.

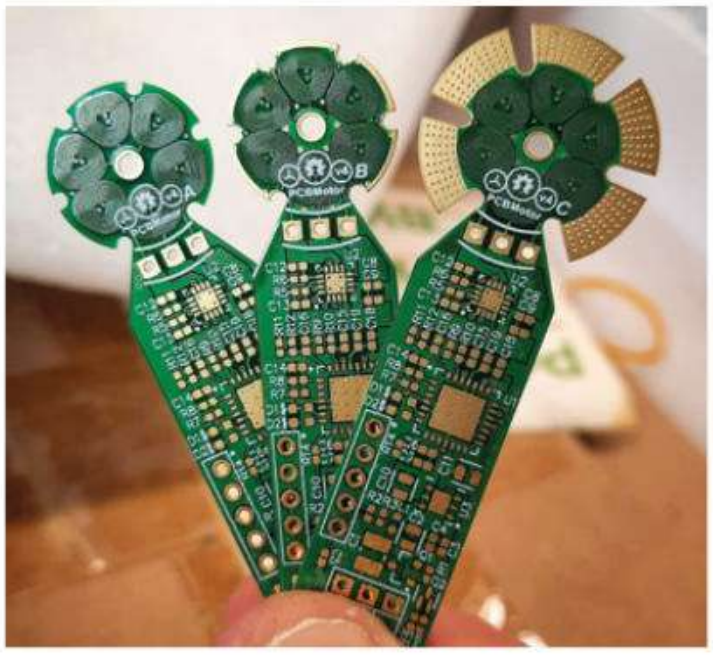
"And then, obviously, it has to do with how strong you can make it in such a small area. So that was my challenge over the years: keep it small and low in weight while still making it strong enough to be useful in, for example, a robot. I can make a larger version that is much stronger, but it sort of defeats the idea of why I started.

"So I've seen that the motor quite low, well, at least version one anyway, was quite low torque.

"You could make the whole thing larger – that would definitely be stronger. I recently made a twelve-layer PCB coiled motor. The improvements with that one compared to, for example, a two-layered version, were remarkable. But then, obviously, you have to keep in mind why PCB motors are good compared to other motor technologies. A twelve-layer board, with the increased amount of copper, is more expensive than a two-layer board. Exploring this stuff is interesting, but you have to remember that the point of PCB motors is to be small and cheap.

"Making bigger motors would definitely improve the torque but, like I said, my main goal for this project was to create a small motor that could be used for drones or robots. The prototype I'm currently developing is going to end up in a wheeled PCB robot.

"The first prototype I made, I wasn't sure that the wheels would have enough traction to move the robot forward. Obviously, there are other issues related to the locomotion of the robot. But I'm now working on a new version, which I think will be one of the coolest projects I'm making this year. It's going to have four wheels, so the robot will also have space for a larger battery. Because when you make small robots, like these, the batteries are a huge problem, because it's very hard to find batteries that are small enough that have high-end capacity. So I sort of made space for a larger battery. I added two more wheels, so the robot will have four wheels. And I managed to do this in such a way that the robot will be a cube – it will be made from a flexible PCB that folds into a cube.



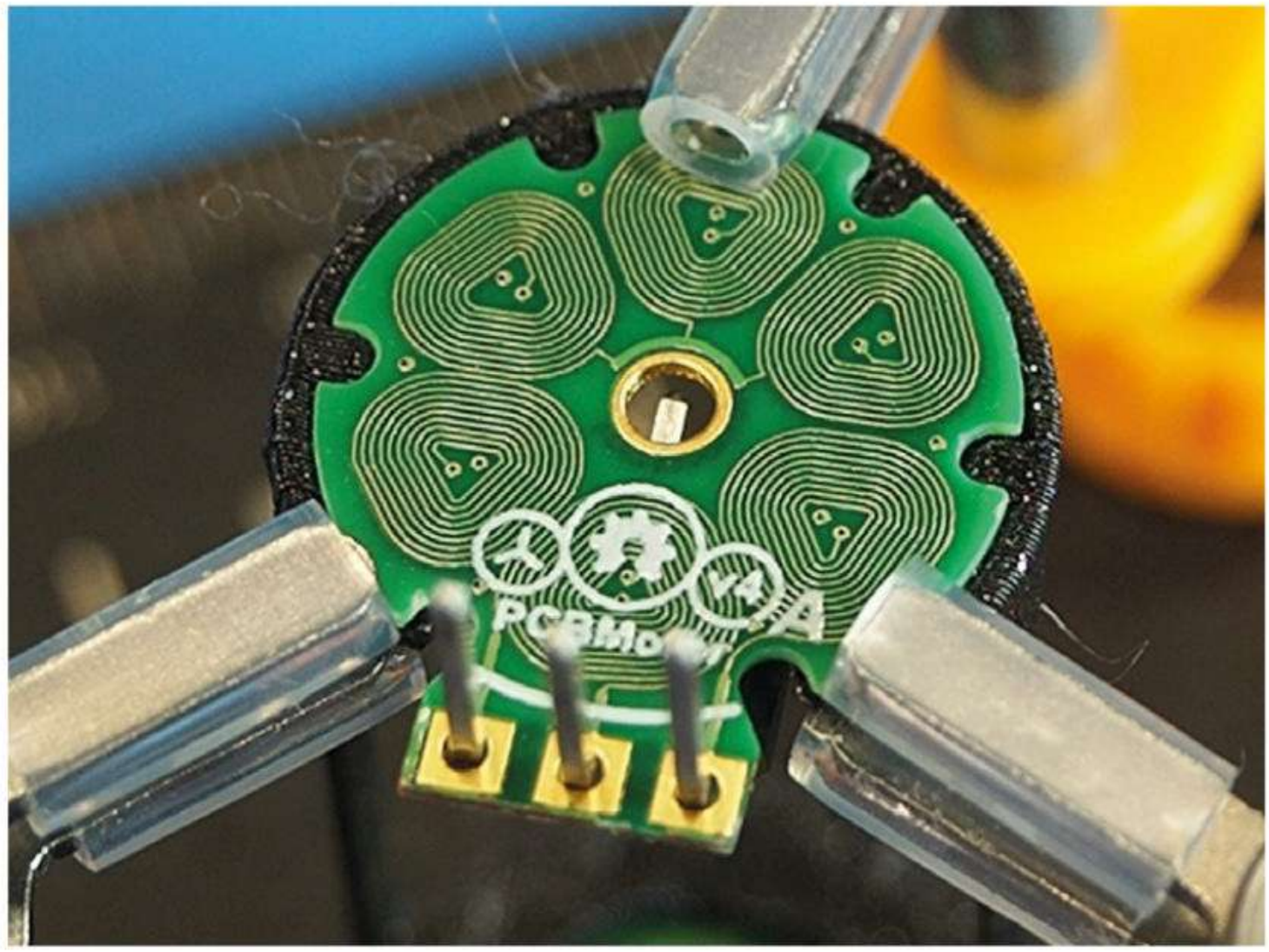
> My main goal for this project was **to create a small motor that could be used for drones or robots**

"So my projects started with the PCB motor, then the second project that used PCBs in a creative way was a linear actuator. And then I started thinking, like, what if, instead of moving the magnet, you could move the PCB? That's when I came up with the idea of using a flexible PCB.

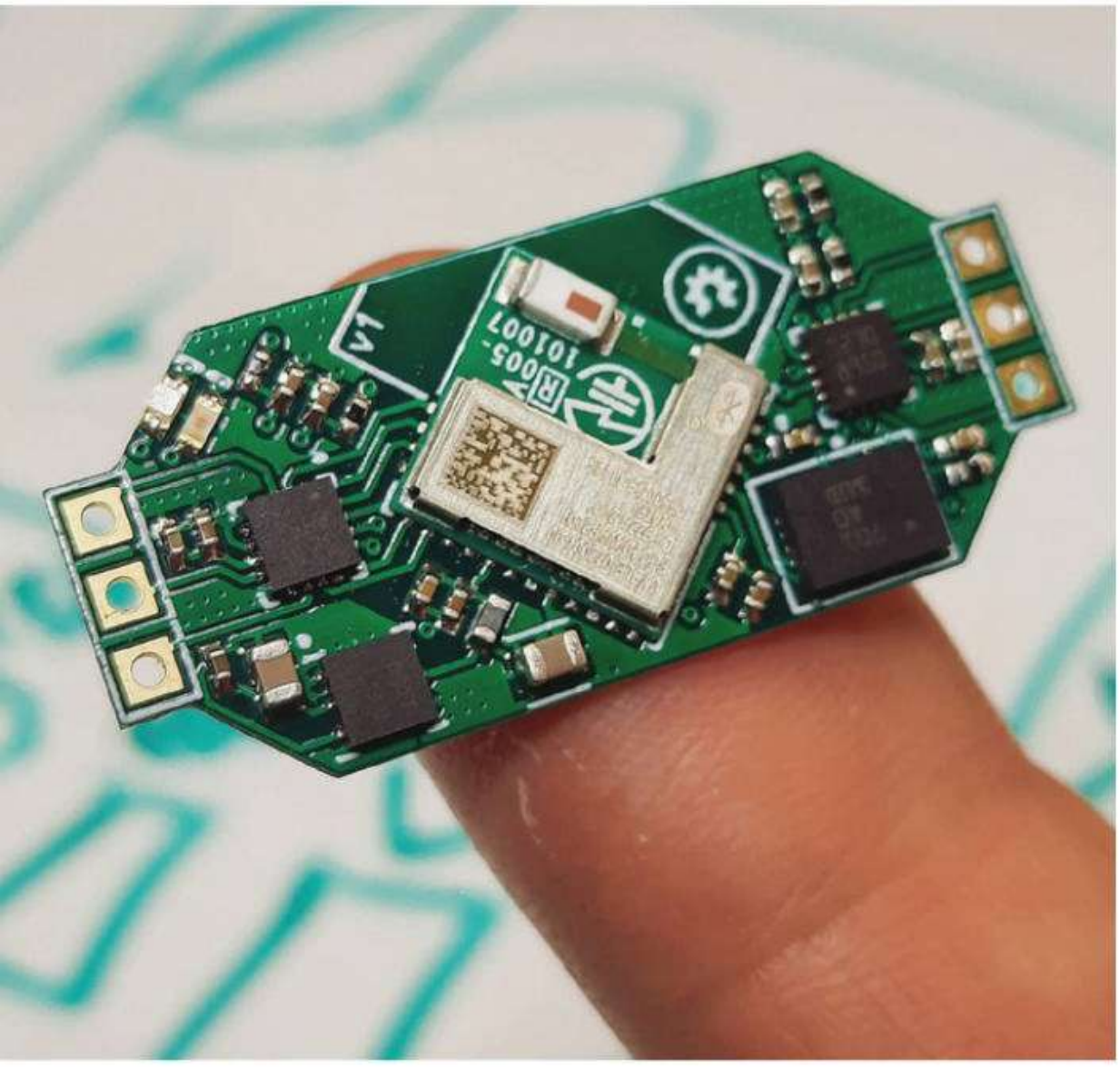
"I had no experience with using flexible PCBs. And so, my YouTube audience saw me grow using flexible PCBs and the mistakes I made with my flexible PCB project because I think it's not that →

Below
By embedding the coils in the PCBs, Carl can keep the motors small and cheap

REGULAR



Right Soldering up a PCB motor

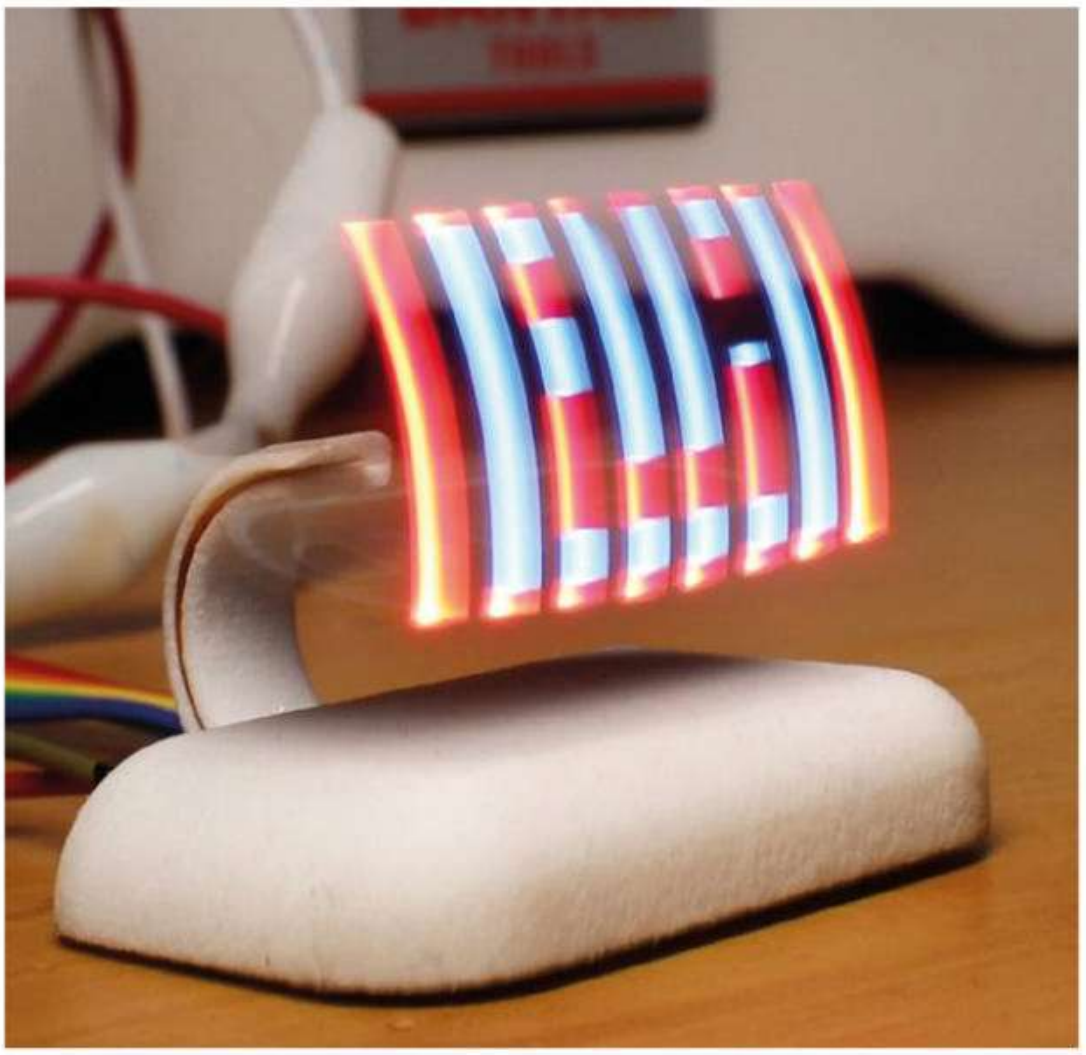


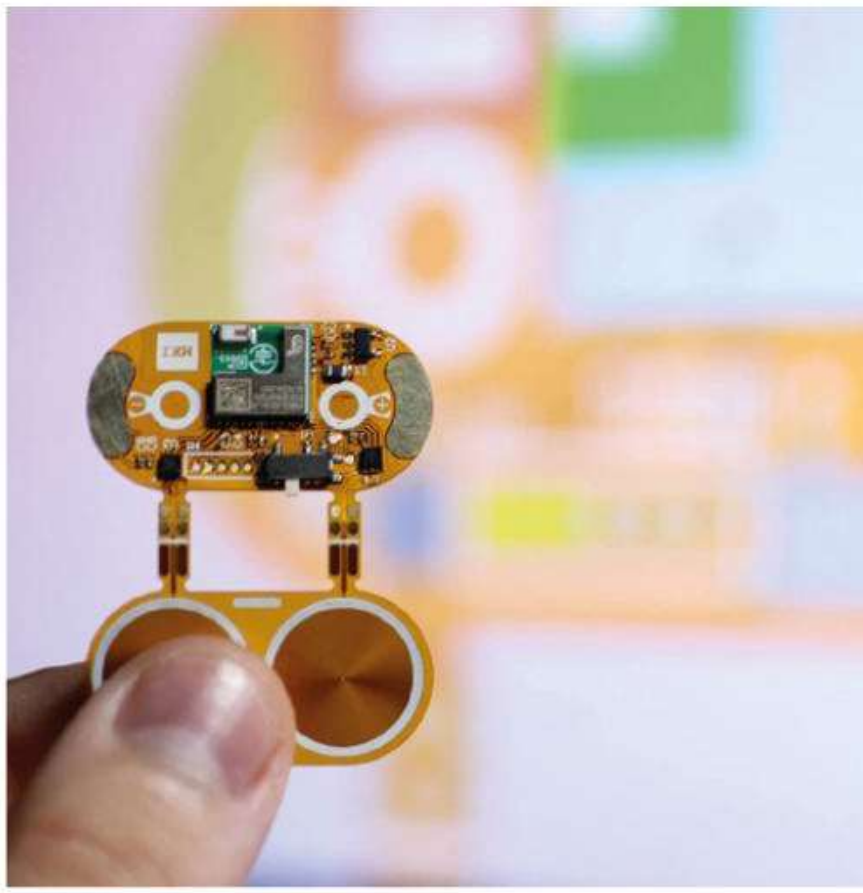
common to use flexible PCBs; it's still a new technology. It's helpful to share videos and documentation with the world, especially on Hackaday. Your videos will be targeted to engineers. I mean, you could inspire someone to make something new, and you'll also get your work in front of people who have experience in specific fields, who will also help you, and perhaps give you some suggestions – that's been very helpful. I don't have a huge experience with being an engineer – I've only officially been an engineer for three years.

"It's also different working with a company and, like, sort of inventing stuff with a team of engineers. I'm currently working self-employed full-time. I used to work in an automotive company, working with electronics and writing firmware for some parts that go into cars. So that's what I used to do before YouTube.

"As a kid, I used to be fascinated with building robots and bringing inventions to life. Obviously, we didn't have the access to learning materials that the kids today have. Nowadays, with Arduinos and Raspberry Pis and YouTube, it's very easy to learn, but before that, it was very difficult to search for something you'd like to learn about and learn it.

"For me, it was mainly just books and maybe television – *MythBusters*, for example. But I mean, this is what I wanted to do from a young age, so I'm very happy that I'm getting to pursue my dreams.





"I've managed to rotate a drone propeller, but not at the high speeds that you would need to achieve lift-off. I need to double-check what the state of the motor is, improve the torque, but I still don't think it's high enough yet. But for now, I'm focusing on the wheeled robot; then we'll see.

"I've also done a fish robot. It was a prototype when I started designing flexible PCBs. My mind started saying, like, wouldn't it be cool if this was adapted to power a fish? And I think it would, because the way that a flexible PCB actuates, it's very similar to the way a fish moves. I think it would be cool, but I don't know if it would be practical – water and electronics don't tend to go together that well. You'd have to keep it super-light, while still protecting the electronics from water, so it would be a challenge. I might have a go in the future, but for now, I don't have any plans.

"I think the best way to learn is by failing. Because when something fails, your mind will start thinking of a hundred reasons why it's failing, and only one of them will turn out to be the source of the problem. So you would probably be testing other ways why it might be failing, which, if it didn't fail, you won't be testing. Failure is a very important part of the process. Obviously, it's not enjoyable to have something fail, especially when you have deadlines and time pressures, stuff like that. But I do try and show it in my videos.

> When something fails, **your mind will start thinking of a hundred reasons why it's failing**

"It's good for the viewers to see failure; it's good to show that building things and making them work isn't easy, that you will get things wrong first time, and that's OK.

"If you're not someone who designs electronics, it's not easy to understand this, but it's part of the process. When you write code, for example, it's taken for granted that you'll not get it right first time; there's always debugging involved in writing software. Failure and building on that failure is what makes the final product looks great."

Above
Using flexible PCBs, Carl made a PoV (Persistence of Vision) display

Letters

ATTENTION ALL MAKERS!

If you have something you'd like to get off your chest (or even throw a word of praise in our direction) let us know at **hsmag.cc/hello**

LASER CUTTER

So, you're building a laser cutter? I hope you got your writer to hand in all the parts in the series in advance, because he's probably going to end up setting himself on fire if he's not paying attention. (This is why I am no longer allowed to keep a laser cutter at home.)

Dev
Manchester

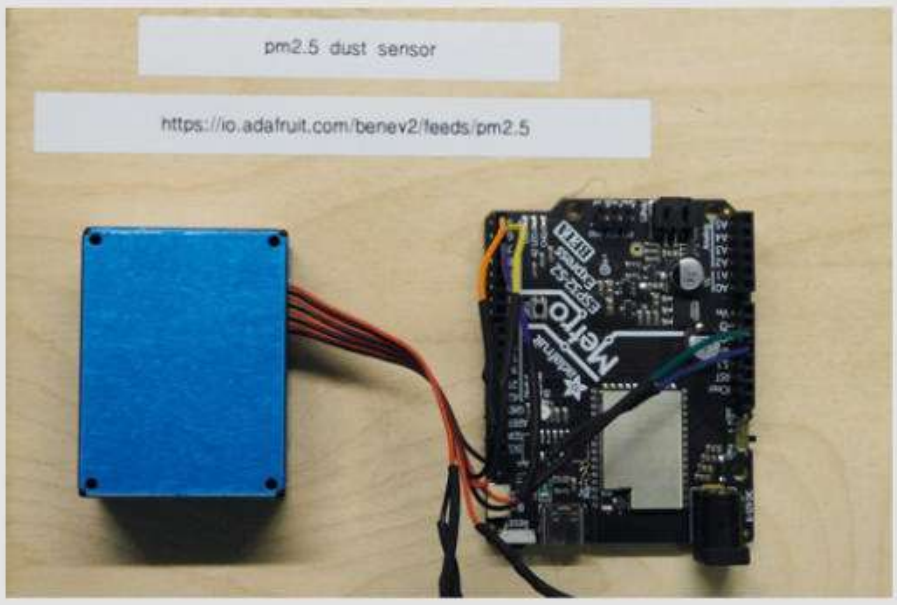
Ben says: Don't worry, Andrew is perfectly safe around lasers and sharp things. He's conversant with all relevant safety guidelines, such as rule 0 (do not be on fire) and rule 1 (do not look at laser with remaining eye). He'll be fine. And actually, if you're following the tutorials, you'll note that one of the benefits of building your own K40-style laser cutter is that you can fix the design flaws that the manufacturers of such units sometimes see fit not to fix. Building one yourself means that you can get a better, cheaper, and safer unit.

PICO HOT WHEELS TIMER

Your writer thought he was building a racetrack timer, but did it ever occur to him that he was also building a science experiment? What is it about one car that makes it go faster? Is it the size of the wheels, the aerodynamics, the weight, or something else? All these can now be accurately assessed! Imagine if Galileo had had one of these when he was throwing things off the Tower of Pisa!

Ciara
Dún Laoghaire

Ben says: We're always amazed by the brilliant things people can do with a couple of sensors, a Pico, and a bit of imagination. If you're putting off your next project build, don't wait: get stuck in!



DUST

The how-to on workshop dust was an eye-opener. I know that some materials, like MDF, produce toxic dust, so I always wear a mask when I'm cutting that (and I only ever cut it when it's sunny enough to work outside). But I must admit I've been a bit blasé about almost everything else. Covid may be on its way out, but it's going to leave us with more awareness of air quality, even if it's just keeping an eye on CO_2 levels and having the window open. Thanks for the pointers, and I'll be setting up my own air quality monitor, to be on the safe side.

Jonathan
London

Ben says: There's no 'safe' level of particulate matter for our lungs, especially when you're working with power saws, solder, the chemicals used in resin printing, or any number of things. The monitor we built satisfied our curiosity, but it's really there as a reminder to open a window and wear a mask (and save up for an extractor unit).

PICO POMODORO

All I ever wanted in life was a tiny lump of plastic and metal to shout at me when it's time to have a break/get back to work. Imagine my delight at seeing Rob Miles's guide to building a Pomodoro timer that'll do just that. Now I need to see if I can get Brian Blessed to record some audio samples for me: "GET BACK TO WORK!", "HAVE A CUP OF TEA!", "HAWKMEN, DIVE!" etc.

Russell
Oxfordshire

Ben says: Well, the great thing about open hardware is that you can modify it to suit your needs. I find that when I'm sitting at my desk with work in front of me, a bell going off is enough to remind me that I have to do something else, but if you need the Shakespearean tones of Mexborough's finest actor to inspire you, you go for it. While we're here, we should point out that this was a project built on the Raspberry Pi Pico, and if you need more inspiration for things to build with this excellent microcontroller, you should turn to page 32.

REGULAR

CROWDFUNDING NOW

Sensor Watch

The retro, smart timepiece

From $35 (not including donor watch) | **hsmag.cc/sensorwatch** | **Delivery: September 2022**

If you're of a certain age (particularly one that sociologists rather disparagingly call 'geriatric millennials'), you'll almost certainly remember the Casio F-91W watch.

Maybe you had one (this author did) but, even if you didn't, you'll probably instantly recognise it. In fact, it was so popular that even 33 years after its release, it's still in production today.

However, while the design is iconic, the features are a little lacking in the modern world. That's where the Sensor Watch comes in. It uses the outer casing and LCD of the original Casio F-91W, but replaces the internals with a modern microcontroller (the SAM L22). This gives you programmatic control over every feature of the watch.

The form factor is much smaller than most smartwatches, and it's based around the legacy LCD. This does bring in a few limitations. Firstly, you can't just put whatever graphics you like on the screen – you're bound by the segments available – the digits are seven-segment, and there are some additional bits that were originally for displaying days, alarm status, etc. You can use these however you like, but you can't add new features to the LCD.

Secondly, there's very little space inside the watch for hardware. There's the basic control board and an expansion slot for a sensor board. The watch comes with a temperature sensor, but you could switch this out for a different sensor, provided you can make it fit. You're unlikely to fit more than one sensor in the space.

> **It's much smaller than most alternatives,** and its retro styling will appeal to many

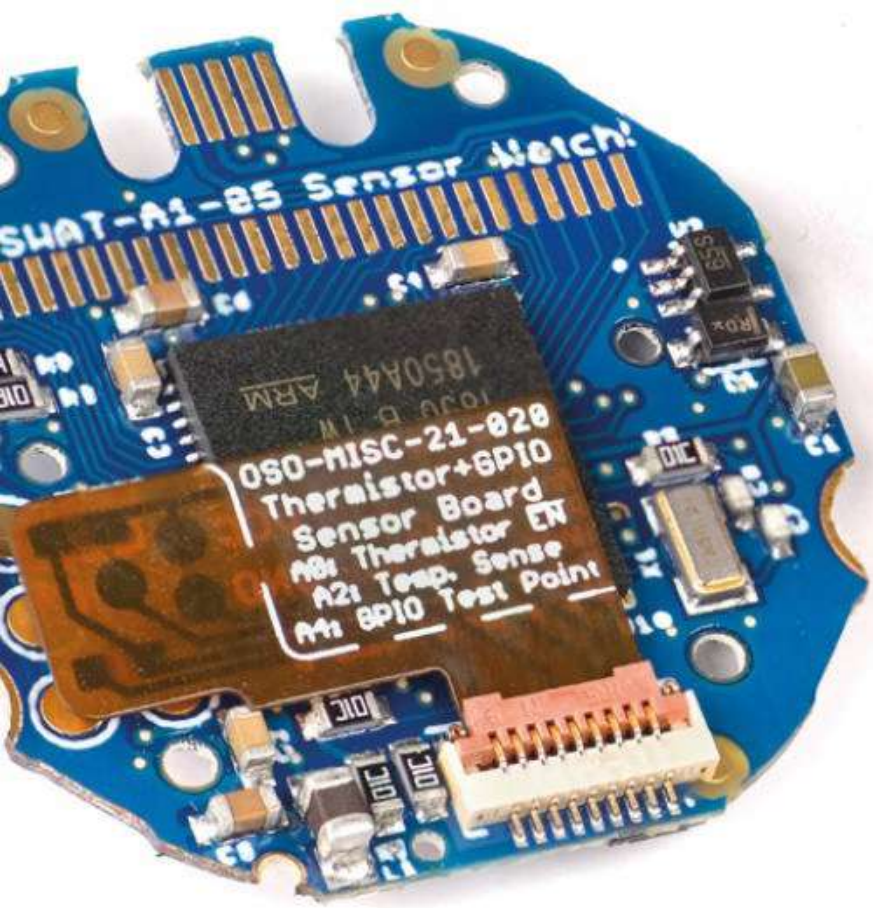
Programming the watch is done through the GNU toolchain. This is slightly more involved than the Arduino IDE, but there are examples to get you started.

We're big fans of hackable smartwatches here at HackSpace mag towers. The Sensor Watch fills an important niche within this. It's much smaller than most alternatives, and its retro styling will appeal to many.

BUYER BEWARE

When backing a crowdfunding campaign, you are not purchasing a finished product, but supporting a project working on something new. There is a very real chance that the product will never ship and you'll lose your money. It's a great way to support projects you like and get some cheap hardware in the process, but if you use it purely as a chance to snag cheap stuff, you may find that you get burned.

Below
This 90s classic gets a 2022 makeover



Where the World Innovates

0800 587 0991
DIGIKEY.CO.UK

Digi-Key® ELECTRONICS

10.7 MILLION+ PRODUCTS ONLINE | 2,200+ INDUSTRY-LEADING SUPPLIERS | 100% FRANCHISED DISTRIBUTOR

*A shipping charge of £12.00 will be billed on all orders of less than £33.00. A shipping charge of $18.00 USD will be billed on all orders of less than $50.00 USD. All orders are shipped via UPS, Federal Express, or DHL for delivery within 1-3 days (dependent on final destination). No handling fees. All prices are in British pound sterling or United States dollar. Digi-Key is a franchised distributor for all supplier partners. New products added daily. Digi-Key and Digi-Key Electronics are registered trademarks of Digi-Key Electronics in the U.S. and other countries.

ECIA MEMBER
Supporting The Authorized Channel

HackSpace
TECHNOLOGY IN YOUR HANDS

LENS

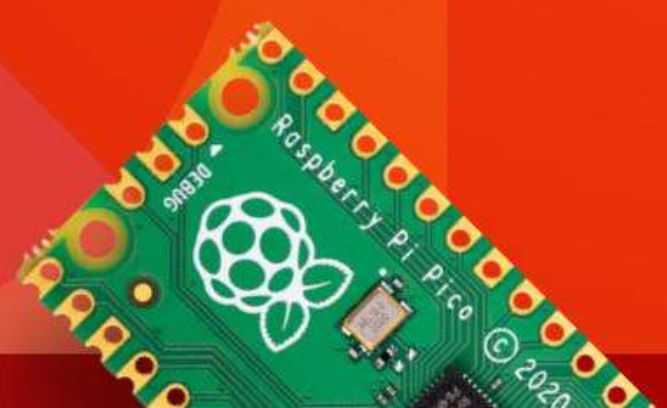
HACK | MAKE | BUILD | CREATE

Uncover the technology that's powering the future

FEATURE

BEST RASPBERRY PI PICO PROJECTS

It's a mighty microcontroller that plays well with other electronics. Rosie Hattersley salutes some amazing Raspberry Pi Pico inventions



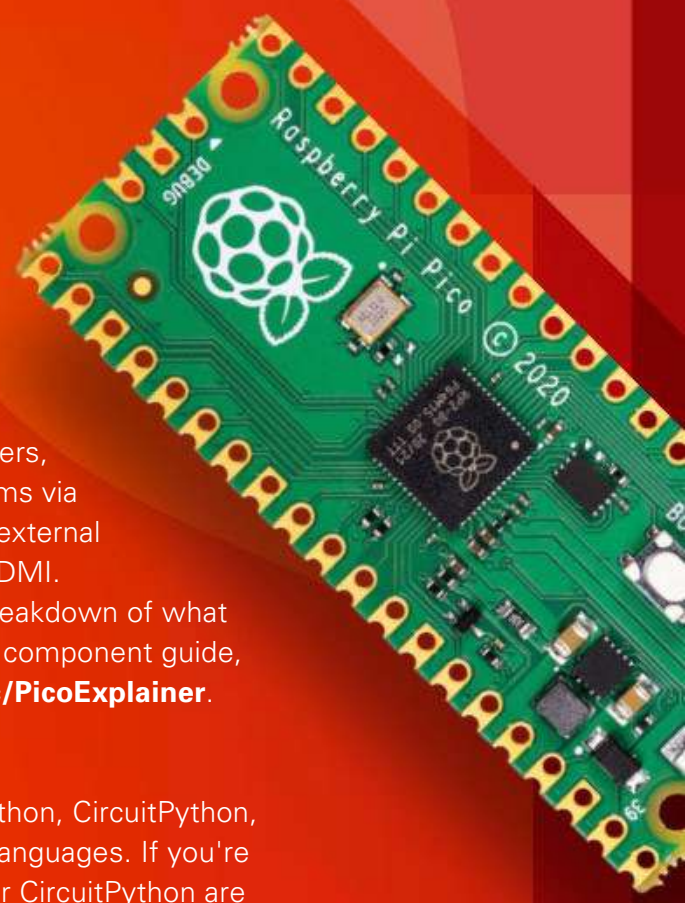
At the end of January 2021, Raspberry Pi took many tech makers by surprise with the launch of the Pico, its first microcontroller, given away exclusively with HackSpace, issue 39. At the time of the Pico's unveiling, there was plenty of excitement about the launch of a brand-new tech toy to keep us entertained and inspire us to get making during the long months of lockdown. There was also both bemusement and approval for the introduction of the RP2040 chip. As well as powering the Pico, the RP2040 was made available as a bare-bones chip from Raspberry Pi, and as part of microcontroller offerings from other companies: Adafruit, SparkFun, Pimoroni, and others.

This equitable approach to giving users options over their choice of compatible hardware can lead to a little confusion when shopping for parts for an exciting new build, but it also means the possibilities of creating projects using Pico and its chip are incredibly broad. The maker community has taken the Pico to its heart, as evidenced by the sheer number of creations built around Pico and the RP2040 chip. A year on from the Pico's launch, we survey the maker scene that has grown up around the tiniest Raspberry Pi, and champion the ways it has come to be used, and the many brilliant projects and makers doing great things with this capable slice of silicon.

First, let's clear up the issue of Pico's form factor, its connections and pins, and the fact it's a microcontroller, not a computer in its own right – and the implications of all that. Whereas Raspberry Pi is a single-board computer capable of multiple tasks simultaneously, Pico is a microcontroller designed to do a specified task and, often, be left alone to get on with doing it. Unlike other Raspberry Pi models, Pico doesn't run an operating system from an SD card. While it still boasts 26 GPIOs – some of them analogue inputs – for connecting other electronics, they have a different layout.

If you're familiar enough with Raspberry Pi to know add-on HATs (Hardware Attached on Top) – modules which provide extra functions such as LCDs that can be used as clocks, game displays, and for displaying messages – you might think Pico loses out on these features. In fact, a third-party carrier board allows it to make use of standard Raspberry Pi HATs, while its own 40-pin layout can be used to attach buzzers, lights, sensors, and to connect to items via jumpers. You can attach a keyboard, external screen, and so on too via USB and HDMI.

If you're interested in a detailed breakdown of what the Pico can do and a component by component guide, there's a great video here: **hsmag.cc/PicoExplainer**.



CONFLICTED ABOUT CODE

You can program Pico using MicroPython, CircuitPython, C, C++, Arduino or a range of other languages. If you're new to programming, MicroPython or CircuitPython are great places to start.

The Raspberry Pi website is packed full of useful resources, including a comprehensive, but simply presented, *Getting started with Raspberry Pi Pico* guide. Taking you through the essentials of programming Pico with MicroPython, this guide covers using Pico with LEDs and how to use PWM (pulse-width modulation) to control the amount of light emitted rather than simply specifying an 'on' state for your lights. Should you want some more coding guidance, we can also thoroughly recommend the official book, *Get Started with MicroPython on Raspberry Pi Pico* (£10 or free to download as a PDF), co-written by HackSpace's very own editor, Ben Everard: **hsmag.cc/MicropythonPico**

ENHANCE YOUR HOME

Wondering how you might make use of Pico around the home? Wonder no longer: it's ideal for smart home and home automation tasks, and has already been embraced by those keen to make use of its monitoring and motion detection abilities.

WIFI IOT

> **hsmag.cc/WiFiIOT**

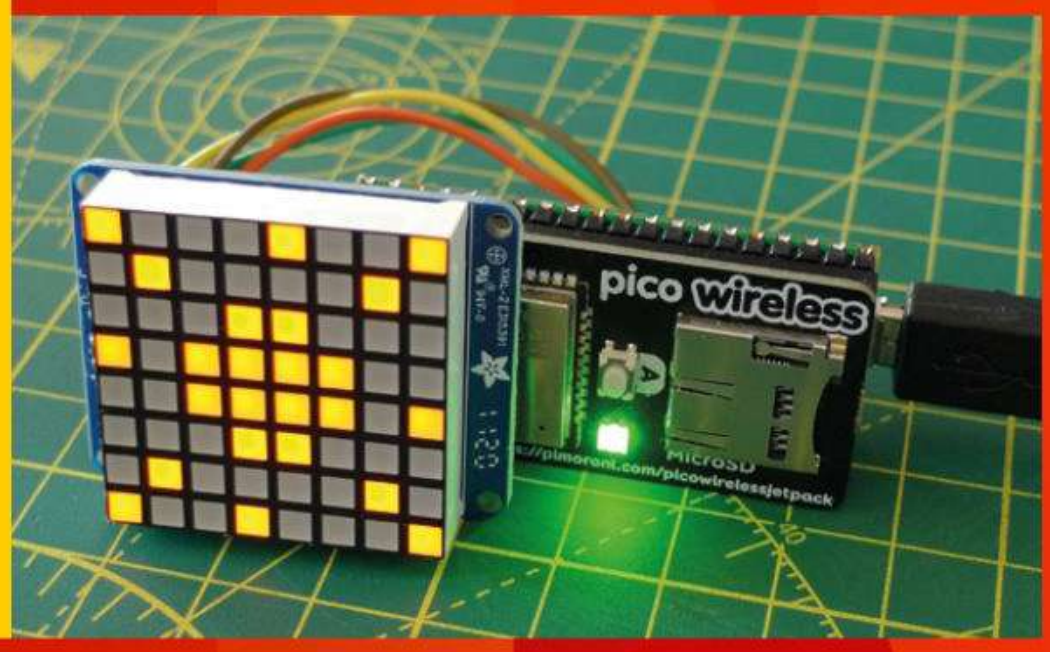
Tony Smith (aka smittytone) recognised the IoT potential of the RP2040 and experimented with both Arduino and Pico versions of a weather display that pulled data from an API and then displays weather info on an 8 × 8 matrix LCD. Using a £12 Pimoroni Pico Wireless Pack, which adds ESP32 wireless connectivity over the 2.4GHz spectrum (plus a microSD card slot, button, and LED), Tony used CircuitPython and the OpenWeather API. He warns that this API limits the number of calls that can be made to it in a 24-hour period but, once up and running, setting Pico to get an OpenWeather update every 15 minutes works fine.

BRIGHTER LIGHTING FOR LESS

> **hsmag.cc/PicoBrighter**

Once you know the basics of how Pico works, HackSpace's Ben Everard has an excellent hack for using it to power lots of LEDs at once, as many as 26 strips of pretty NeoPixel strips at once, in fact. Using a "neat little trick Pico has up its sleeve", Ben explains how to use PIO (programmable input-output) to send data to the NeoPixel strips using very little of the Pico's CPU power. It's an adaptation of the PicoSDK and could have lots of other useful IoT applications – one commenter used a similar idea to have lights attached to a landline flash when a call comes in – though the blazing lights effect is pretty stunning!

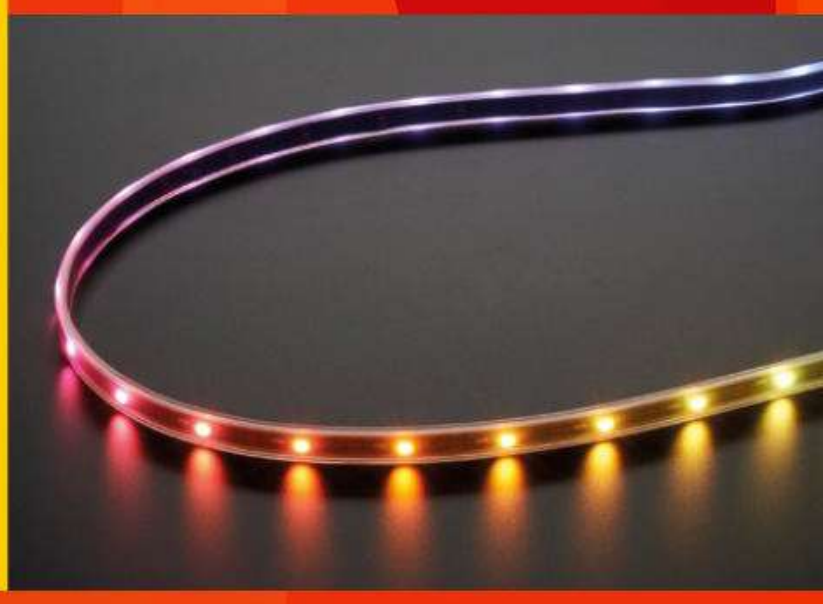
IT'S IDEAL FOR SMART HOME AND HOME AUTOMATION TASKS



PICO ELECTRONIC CLOCK

> **hsmag.cc/PicoClock**

For a fairly simple introduction to using Raspberry Pi Pico, the Waveshare Pico clock kit is a good shout. As well as an LED real-time clock to build, there's a selection of buttons and buzzers, plus a temperature sensor. Full instructions are provided for using MicroPython and C++ to program your Pico clock.

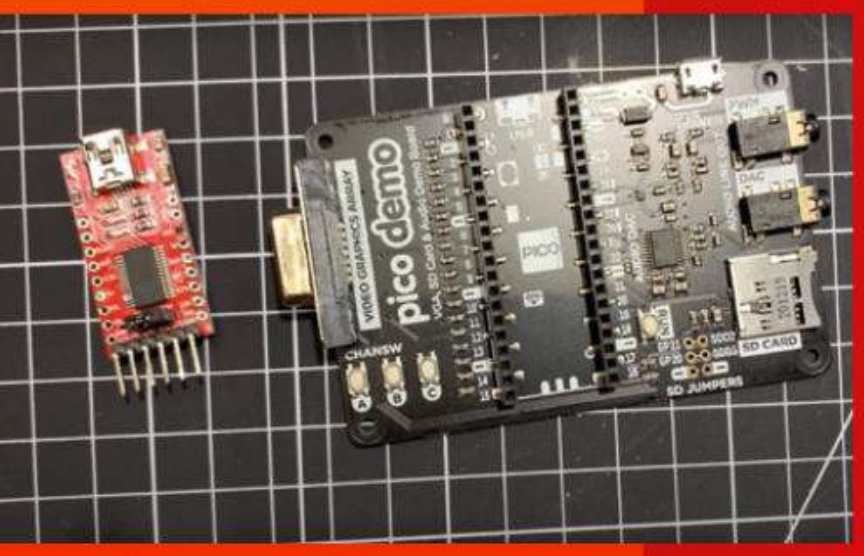


GET TO GRIPS WITH THE LED LIGHTING

> **hsmag.cc/PICO-LED-Lighting**

The reassuringly gentle guide to *Getting started with Raspberry Pi Pico* takes you through the process of using it to control LED lighting, giving a thorough grounding in Pico's abilities along the way.

GAMING



BBC MICRO EMULATOR

> **hsmag.cc/Pico-Micro-Emulator**

BBC Micro B fan and owner Robin Grosset makes use of a BBC Basic emulator ported for Pico (see **hsmag.cc/kilograham**), Pimoroni's Pico VGA Demo Base board to connect a screen, and an audio output so he can enjoy authentic 1980s game sounds while playing his favourite BBC Micro game, Phoenix. Robin then connects the Pico setup to his MacBook, as well as using a 3.3V serial adapter to connect the keyboard of his BBC emulator. He then downloads specially written BBC Micro emulation code by Graham Sanderson (choose the RP2040 option if you try this project yourself), where there is also code for the event forwarder that allows you to control the game via the keyboard. The Phoenix game is a version written in 2017 as a retro project and downloaded from **bbcmicro.co.uk** with controls mapped to a modern QWERTY keyboard – Robin shows off the slightly different key arrangement on his actual BBC Micro B.

The gaming experience is pseudo-authentic: as Robin's YouTube video explains, the background sound when he 'loads' Phoenix on his Pico is that of it seemingly accessing its absent floppy disk!

For those keen to see other 1980s games for the Pico, Robin promises future ZX Spectrum and Commodore 64 emulation videos to come.

IT'S SUPER-EASY TO CODE FOR, IT'S REALLY CHEAP, AND IT BOOTS UP IMMEDIATELY

RETRO TECH

JOKER PHONE

> **hsmag.cc/Pico-Joker-Phone**

Pico might be the newest Raspberry Pi model, but it's got plenty to offer upcycling and retro-hacking fans. Team DIY Champs took it upon themselves to update a rotary dial telephone in order to bring a bit of light-hearted nonsense to proceedings in the form of groan-worthy Dad Jokes (see the API at **icanhazdadjoke.com**). The Pico Phone project uses a custom PCB with Notecard and a SparkFun 12-button Qwiic Keypad, along with some CircuitPython code and a Twilio account to connect to the Dad Jokes API. Once assembled, dial the Pico Phone's number, hold the handset to your ear, and press the # key to hear a terrible pun.

ARCADE-STYLE REACTION GAME

> **hsmag.cc/PicoLightArcade**

If, for you, a day out at the seaside is incomplete without a trip to the arcade for a game of air hockey or another means of expending energy arbitrarily/skilfully bashing a puck or flipper, you'll probably like this reaction game. Based on Mogura Taiji, a Japanese form of Whac-A-Mole, it challenges you to bash each lit LED in turn, a bit like Simon or one of those annoying dance mat challenges, but without the awkward disco moves.

Maker Thomas Roth's version involves arcade buttons with holes for 20 of them drilled into a small wooden tabletop console. Having hooked these up to the GPIO pins on his Raspberry Pi Pico in a 3 × 7 matrix, Thomas designed the game for two people to play at once, controlling nine LEDs each. He also includes Start and Mode buttons in the middle of the board. Pico seemed an obvious choice to control everything: "Firmware updates are drag and drop, it's super-easy to code for, it's really cheap, and it boots up immediately," he explains. For the Whac-A-Mole-style game, players try to press the most buttons in a 45-second burst, while Thomas's second game option is a take on the aforementioned Simon sequence memorising challenge. He aims to make more games for his DIY arcade and has already made an online scorekeeper, but he loves the idea of a seven-segment display connected to his homemade games machine. →

GREEN SHOOTS

SOIL MOISTURE SENSOR

> hsmag.cc/PicoSoilMoisture

Andy Warburton knows just how much houseplants are at the mercy of humans – so much so that he made a gorgeous soil moisture sensor that not only alerts him to overly dried out or waterlogged soil, but looks striking too. Adding a SparkFun Soil Moisture Sensor, a USB connector cable, and a NeoPixel ring of lights to a fairly straightforward sensor setup – using Adafruit's CircuitPython library and his own downloadable code – results in an eye-catching setup. The initially blue NeoPixel lights then need to be calibrated, after which it's ready to start looking after your little Monsteras.

IT'S GETTING HOT IN HERE

> hsmag.cc/Pico-Temp-Humidity

Whether you're planning on a humidor to look after your cigars or a thermal and moisture-controlled environment for your reptiles or tropical plants, Pico is ideal for monitoring the humidity of an enclosed environment and keeping tabs on changes over time. The £4 DHT11 humidity sensor can be bought individually, or comes as part of the Waveshare starter kit mentioned in the burglar alarm project (page 37).

IT HAS JUMPER WIRES, RESISTORS, AND SWITCHES

KITS AND EXPANSION BOARDS

Several companies sell Pico kits to help you get straight into exploring the making aspect of using the microcontroller.

Kitronik's Discovery Kit comes either with or without a Pico, and offers components to try seven experiments. As well as myriad sensors and buzzers, it has jumper wires, resistors, and switches, plus a detailed guide on what to place where on the included large breadboard: **hsmag.cc/PicoDiscoveryKit**.

OKdo offers the same Kitronik kit, but also has a more basic £7 version with USB cable, jumpers, and headers, but lacking the buzzers and lights: **hsmag.cc/OKDOPico**.

Arducam Camera Module For Pico
If you've a mind to make use of Pico's compact dimensions for some surveillance or wildlife watching, Arducam's dedicated camera module is an obvious choice – no expansion kit required: **hsmag.cc/Arducam**.

Waveshare GPIO Expander HAT
One of the features that makes other Raspberry Pi models so adaptable is the number of different HATs (Hardware Attached on Top) that are available to use with them. As the acronym suggests, these neat little extras sit atop your Raspberry Pi and can be straightforwardly swapped for a different HAT with different features. Pico's GPIO layout is different, so standard HATs can't be attached. However, Waveshare's £7.50 GPIO Expander board provides a standard Raspberry Pi 40-pin header, so you can use Raspberry Pi HATs with your Pico too, and also adds a 40-pin Pico header designed to take LCD and OLED screens, further expanding your project options: **hsmag.cc/WaveshareGPIOExpander**.

The Pi Hut Pico HAT Expansion
One of several similar boards that allows you to use Raspberry Pi HATs with your Pico, much like the Waveshare version described above: **hsmag.cc/PicoHATExpansion**.

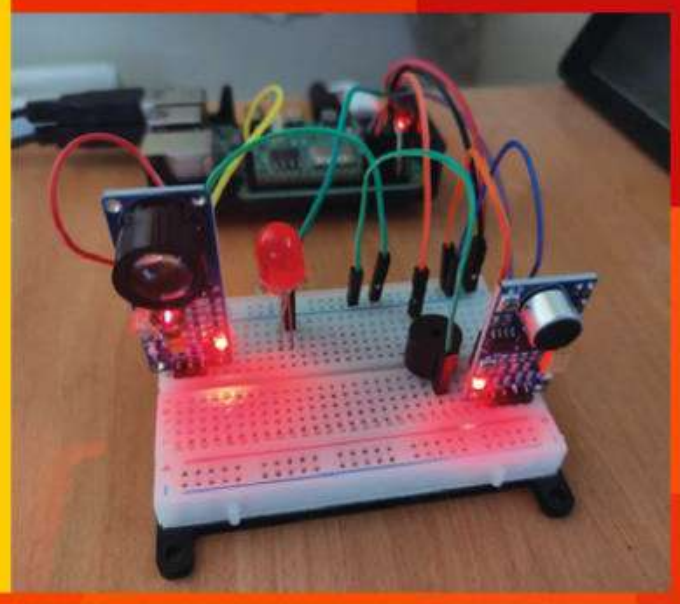
Pimoroni Pico DV Demo Base
Handy development board for users of C++ with some knowledge of using microcontrollers: **hsmag.cc/PimoroniPicoDVDemoBase**.

STAYING SAFE

STOP, THIEF!

> **hsmag.cc/PicoIntruderAlarm**

If you're worried about burglars and snoops, the sound sensor and laser in Waveshare's Sensors Pack (or a similar sensor kit) will immediately jump out as useful items with which to build an intrusion detection device. Designed to foil even the canniest of cat burglars, the laser-based intruder alarm uses exactly the principle beloved of museums keen to snag would-be jewel thieves. The all-important, not to be interrupted, laser beam is sited such that an intruder is certain to encounter it, but where it's impossible for someone to inadvertently be dealt an eyeful of damaging laser light. Fairly near the floor is the safest bet. Other light sources can't be mistaken for the laser since the detector only responds to light with precisely the same 650nm wavelength. Connect the laser to Raspberry Pi Pico, then use Pico's 3V3 pin to add power. Install the provided code for the laser sensor, and do the same for the sound sensor which triggers a piezo buzzer. Pico's LEDs should light up if everything is correctly installed. Make a loud noise to test it's working.



SCARE OFF INTRUDERS

> **hsmag.cc/Pico-Foxes**

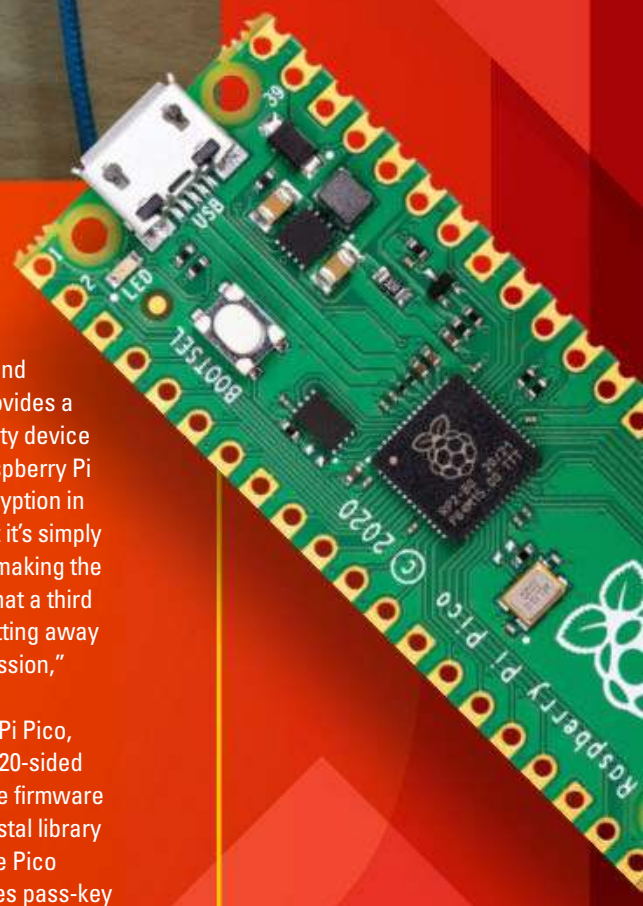
This off-topic discussion regarding telltale signs of foxes visiting, sparked debate on how best to deter unwanted garden visitors, whether using sprays, flashing lights, or unpleasant sonic alarms. Noting that a Raspberry Pi and some AI could potentially be used to ascertain the prowler is indeed a fox, and to discern which tactic to use, we hereby throw out the challenge to come up with the best Heath Robinson device using Pico, or something else, to humanely scare off vulpine visitors.

ENCRYPT YOUR FILES

> **hsmag.cc/Pico-File-Encryption**

If digital theft is more of a concern than breaking and entering, Max Bortnikov's encrypted data vault provides a great use for a Pico. Max is an experienced security device designer, with seven other Instructables using Raspberry Pi or Arduino to his name. His Pico version uses encryption in a bid to make the data owners' files so secure that it's simply more trouble than it's worth to hack into them. By making the effort involved "higher than any possible reward that a third [party] can get by obtaining your data… you're putting away incentives to access your data without your permission," he reasons.

The setup involves an ESP32 board, Raspberry Pi Pico, an LCD screen, three different resistors, oh, and a 20-sided dice, if you can get hold of one. Having updated the firmware and Arduino IDE for ESP32 and Pico, the LiquidCrystal library is installed via Sketch, followed by firmware for the Pico downloaded from **hsmag.cc/Northstrix**. Next comes pass-key generation: throw the dice and follow Max's instructions with regards to using this random number generation technique to create a highly secure key. Update the firmware with the key details, and flash to update the Pico before attaching the LCD and testing the random number generator is working – a screen full of garbage suggests it is. Finally, set a master password which is lost as soon as the Pico is disconnected but is used to unlock it, and ensures someone getting physical access to the device can't simply scroll through your notes.



EVERYDAY LIFE HACKS

BETTER BURGER MAKER

> hsmag.cc/PICO-sous-vide

VEEB knows that food and drink rank high among life's essentials. *MasterChef* fans get to combine tech and foodie bragging rights with a DIY sous vide known as Heat-o-matic that uses a hotter-cooler approach to cooking burgers and other morsels. Key to it all, of course, is a temperature sensor and switches connected to the Pico.

SOLDERING ON

The possibilities afforded by microcontrollers mean there's every chance you'll need to solder items to each other. Soldering irons are great for through-hole components, but if you move on to surface mount things, you might need a bit more control. Hot plates, like this MHP-30, make it easy to solder on even the tiniest components.

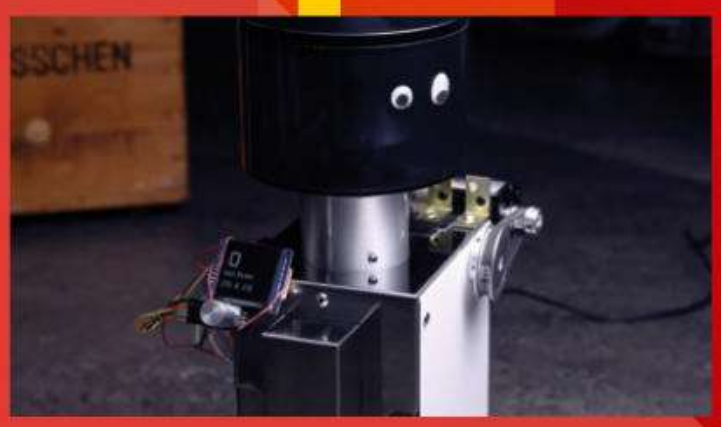
> hsmag.cc/PICO-hot-plate

BESPOKE COFFEE GRINDER

> hsmag.cc/Pico-coffee-grinder

Some people just really, really care about coffee. If you're a bean aficionado, you may well appreciate the ability to have a techie way of switching between grind settings whenever you switch between preparing an espresso from a coffee dripper to a smooth Americano from a cafetière, neatly avoiding the horror of a well-earned cup of cawfee being ruined, ruined, I tell you, by an incorrect grind. VEEB's approach carefully calibrates the exact grind setting you prefer for each coffee preparation method and style of drink you select, then saves it as a preset. The Pico comes into play with a rotary encoder switch and Waveshare DC motor board for Pico to switch between grind settings on VEEB's Bezzera BB005 coffee grinder and in powering the OLED used to select a grind.

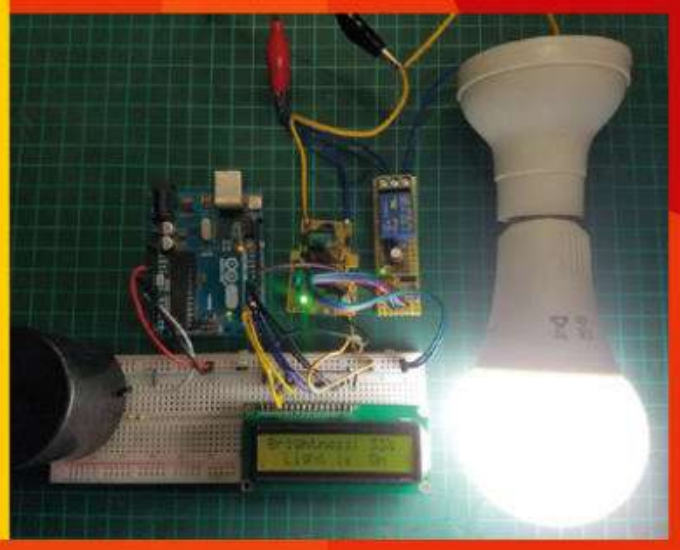
Install MicroPython on your Raspberry Pi Pico before installing the GitHub code (**hsmag.cc/Dailygrind**) on your computer and Pico.



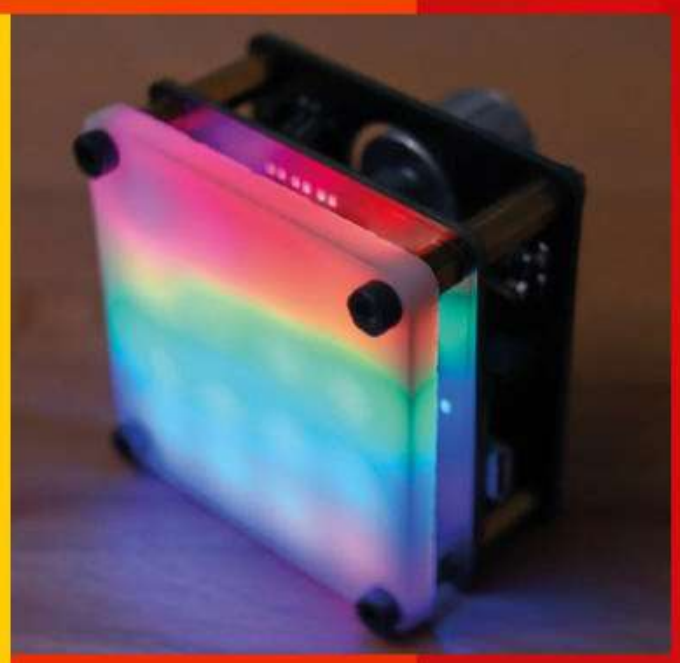
DIY SAD LAMP

> hsmag.cc/Pico-SAD-lamp

If gloomy days and nights get you down, boosting the amount of light around you can lift your mood. Seeed Studio's Grove Shield offers an easy way to connect Raspberry Pi Pico, while Seeed's guide to fun Pico projects includes instructions for a smart bulb that comes on automatically when it gets dark and gradually dims as the amount of daylight increases. The project (adapted from this one for Arduino: **hsmag.cc/Arduino-light-control**) just needs a light sensor with varying resistance to light intensity. To turn it into a SAD (seasonal affective disorder) lamp, add a power source and some form of display with or without a translucent shade to make it realistically lamp-like.



SHOW OFF YOUR WARES



PERFECT PRODUCT SHOTS WITH PICOLIGHT

> **hsmag.cc/Pico-photo-light**

Reflections and shadows (not to say fingerprints) are an absolute bane for anyone who needs to photograph shiny things. The PicoLight makes it much easier to take eye-catching product shots that catch potential customers' eyes on Instagram, Etsy, and elsewhere. Cleverly, the makers pair a Pico PCB with 4×4 NeoPixel matrix light array with a LiPo battery shield that diffuses light. A rotary encoder with push-button control is used to adjust the colour intensity and hue. The LiPo shield PCB also has a power button and charging circuit, although there's also an option to power the PicoLight via USB. M3 standoffs keep the two PCBs separate. Code and further details can be found on maker Zalmotek's GitHub page: **github.com/Zalmotek/PicoLight**.

MAKER PI PICO RFID READER

> **hsmag.cc/Pico-RFID-reader**

Cytron's Maker Pi Pico is essentially a kit version of Raspberry Pi Pico that adds useful extras to the Pico and makes it a bit easier to get to grips with if you're new to using microcontrollers. (Note there is also a Base version of Maker Pi that provides just the add-on pins and ports and doesn't include Raspberry Pi Pico.) Usefully, Seeed Studio offers a freely downloadable guide to getting started with Raspberry Pi Pico and MicroPython: **hsmag.cc/GroveShield**.

The short RFID tag reader YouTube tutorial makes use of a Maker Pi Pico and adds a relay module. The RFID card, or perhaps wristband, contains instructions that are then acted upon (tech resellers have packs of five for £5 or so). While QR codes came into their own during the pandemic, RFID and NFC also have plenty of creative applications should you wish to use your Pico to help users make social connections, tweet or post to Instagram, or follow your charity's runners via the RFID tags on their trainer laces. Or, you could simply give yourself a bit of a glow-up with RFID reactive nails: **hsmag.cc/GlowNails** →

ROTATING PHOTOGRAPHY TURNTABLE

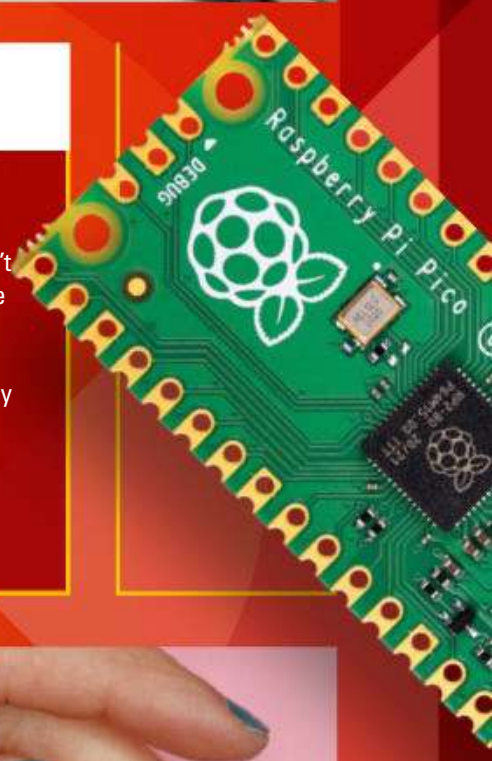
> **hsmag.cc/Pico-Turntable**

As a maker, it's often useful to be able to show off your creations, whether for commercial reasons or simply to display them. This Pico-powered rotating turntable is also ideal for checking your design works from all angles and whether you missed a bit when painting it. Little outlay is required: a stepper motor, Arduino power module and motor driver, lazy susan bearing plate, USB-A to USB-C cable, jumper leads, spacers, screws, a 9V battery, a breadboard, and of course, a Raspberry Pi Pico. The Pico is programmed using the Thonny IDE and MicroPython, while the turntable schematic can be downloaded and then 3D-printed. Maker Will Lawson warns that the resulting turntable is not designed for heavy rotation, but works well for showing off lighter items.

WEARABLES

The diminutive proportions of the Pico seem to cry out for it to be used in wearable projects, but Adafruit and NeoPixel already have a lot of products aimed at costumiers and cosplayers. Nonetheless, we couldn't help but raise a smile when we saw Tom's Hardware's Ash Hill embrace the concept by sporting some suitably raspberry-hued headgear topped off with a 3D-printed Raspberry Pi logo and some flashy LEDs powered by a 3.7V LiPo battery and controlled, of course, by a discreetly worn Pico. Ash's fancy fashion statement sees her Raspberry Pi hat accessory sashay gently through the RGB spectrum via Colorsys's modified Python code. Neat.

> **hsmag.cc/Pico-beret**



KEYPADS AND CONTROLLERS

ARCADE BUTTON MIDI BOX

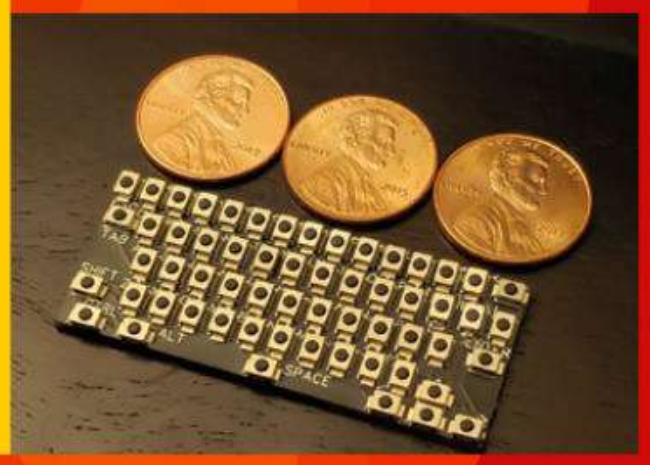
> hsmag.cc/Pico-midi-controller

Adafruit has come up with a great alternative use for arcade buttons: using them as the buttons for a MIDI synthesizer box. A Raspberry Pi Pico sits proudly on show above the pleasingly tactile LED arcade buttons inside the 3D-printed case. The build also features Adafruit's AW9525 GPIO expander and LED driver. An OLED screen and joystick mean the MIDI box can do dual duty as music maker and game controller.

MINI KEYBOARD

> hsmag.cc/Pico-mini-keyboards

Maker TEC.IST took up the challenge of making the tiniest keyboard around, reasoning that he hadn't encountered one "small enough for wearables or extremely small PCs". As TEC.IST reveals on his Hackaday project page, he built the 59-key board to match the size of the Raspberry Pi Pico, with the microcontroller handling the key matrix decoding and USB interface, and designed a custom circuit board to hold everything. There's a full QWERTY layout, plus a number row, mimicking the layout found on computer keyboards. Touch-typing might prove a challenge!



SOUNDS SWEET

HIS TUNING DEVICE LISTENS TO THE SOUND OF A STRING

AUTOMATIC GUITAR TUNER

> hsmag.cc/Pico-guitar-tuner

When Guyrandy Jean-Gilles told fellow Reddit users he'd created an automatic guitar tuner, it took a while to dawn on them that his invention does more than confirm that the strings have been suitably tensioned to avoid offending sensitive ears. In fact, the Pico-controlled tuning box contains a motor that actually tweaks the strings to the exact pitch required. His tuning device listens to the sound of a string being plucked and decides which note it needs to be tuned to. It then turns the tuning keys on the guitar's headstock just the right amount to achieve the correct note. Guyrandy's tuning device (for which he provides 3D-printable files and source code) takes its cue from the Roadie 3 and uses Pico's audio recognition function. The user presses the pitch increase or pitch decrease option and selects a target note, before placing the tuner device on the peg of the string that needs to be tuned. Next, the musician plucks the string and Pico's sensors listen and it calculates the difference between the desired sound and the actual tone emitted. It then turns the peg and displays a 'done' message on the tuning device's screen when the string has been tuned.

Although Guyrandy wasn't fully happy with how effectively the motor turned the guitar pegs, his project was enthusiastically received with several suggestions on enhancements and possible extra features, including alternative tunings. As it stands, the Pico-enabled project can be found at: **hsmag.cc/AutoGuitarTuner.**

ENVIRONMENT

PICO ORRERY

> **hsmag.cc/Pico-orrery**

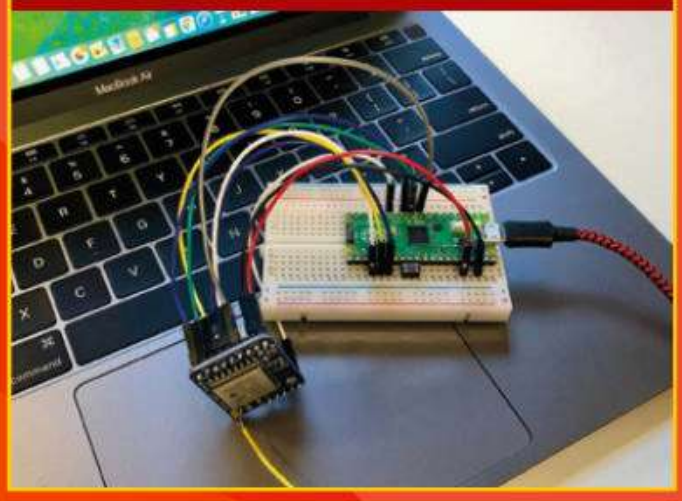
Simulating the movement of the planets around the sun is the sort of creative challenge that early astronomers loved. The orrery dates back to around 150 BCE, but is named after the 4th Earl of Orrery, who commissioned one in 1713. Dmytro Panin has designs on making his own mechanical orrery someday, and has already got as far as creating a digital version. "Astronomy has always fascinated me, and I wanted something that I could look at and see the planets in the solar system progress over time, so I built a visual reminder," he told our colleagues at The MagPi. As well as Raspberry Pi Pico, Dmytro's version has a Waveshare Precision RTC expansion module, since the exact time is critical to all things astronomical. A Pimoroni Pico Display shows off the results of Pico's calculations of which celestial being is where.

If you've little need to check the current position of Saturn, but wouldn't mind knowing the time of day, dr-mod (as Dmytro is known on Reddit) has also come up with a Pico-based true binary clock using some of the same code: **hsmag.cc/Pico-binary-clock**.

LONG-RANGE CONNECTIONS

LoRaWAN (long-range wide area network) has become a go-to means of connecting devices via the cloud and over distances of several kilometres, and has been fully embraced by IoT makers as well as businesses. It's perfect for collecting data from remote locations. There's now a worldwide network of LoRaWAN servers that you can use to connect to, should you be thinking of using Pico to set up an internet-based project: **thethingsnetwork.org**. The link below explains how to go about adding LoRaWAN to your Pico.

> **hsmag.cc/Pico-Lorawan**



BALLOON TRACKER

> **hsmag.cc/Pico-Balloon-Tracker**

Dave Akerman, of High Altitude Ballooning, spends his free time launching and tracking balloons. When he hears about new types of processors, he feels compelled to try them out for his balloon-tracking purposes. Microcontrollers such as Pico are ideal. "Anything with a modest amount of code memory, data memory, processor power, and I/O (serial, SPI etc. depending on choice of GPS and radio) will do," he says. Alongside this, he just needs a radio transmitter and a GPS receiver. Using a prototyping board, he connected up a Pico, added a LoRa radio transmitter via SPI, a u-blox GPS receiver, a BME280 temperature sensor, and an I2C environmental sensor. A small battery powers it all.

Status and environmental data is received on the ground by a Raspberry Pi LoRa receiver and uploaded to an internet database that drives a live Google Map.

SUBSCRIBE TODAY

FOR JUST £10

Get three issues plus a
FREE Raspberry Pi Pico
delivered to your door

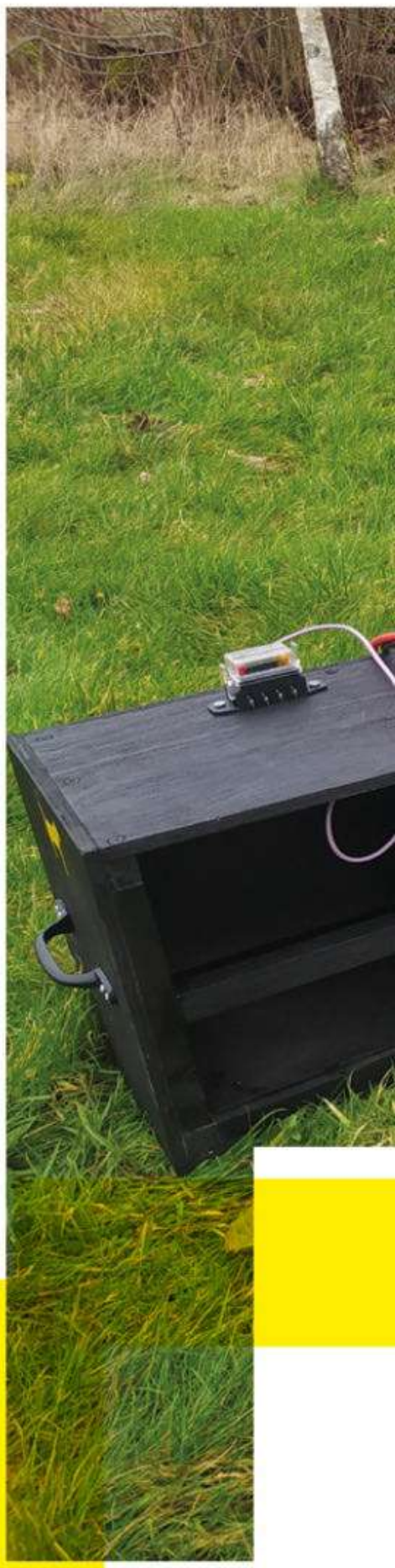
hsmag.cc/FreePico

UK offer only. Not in the UK? Save money and get your issue delivered straight to your door at **hsmag.cc/subscribe**. See page 66 for details.

Subscription will continue quarterly unless cancelled

FEATURE

HOW I MADE



A PORTABLE SOLAR GENERATOR

By Jo Hinchliffe

Small solar power generators for car camping or glamping have become all the rage, but for a fraction of the price, I decided to build my own system that can be expanded over time

Solar panels for 'off-grid' power isn't a new concept, but it's true to say that it's an area where technology is consistently improving and becoming more efficient and reducing in cost. I often consider while working in my shed that the only power requirement needed in there is for lighting, so it would be a suitable candidate for some 12V lighting paired up to a solar power system. The other area I think a solar power system would be useful is when I attend rocketry events where I'm camping for numerous days without an electrical hookup, but with a vehicle to carry my kit. Recently, numerous 'solar generators' have been hitting the market, often around the £500 mark – they consist of a solar panel and a smart-looking case containing some charge control equipment, a battery, and some 12V or possibly inverter-fed 240V outputs.

Above
The complete setup – portable enough to be carried into the field

I decided to build my own semi-portable solar generator – it could live and work in the shed, but also be carried short distances to be set up in a tent or van. My use cases for it when at events are to be able to charge a phone and laptop, and also to be able to supply 12V to my LiPo battery charger, as well as some small amounts of lighting. I would also like to be able to partially recharge my larger 48V electric bike battery, potentially using a 12V to 240V inverter as a quick and simple approach whilst I research a more efficient 12V to 48V conversion system.

With my rough use case in hand, I started to specify components. Essentially we need a solar panel, a battery, and a charge controller, plus a few other items to make the system safer and easier to manage. Looking at solar panels first, it seems that the minimum entry point to this type of system would be a 100W panel. A 100W panel should, on a sunny day in a good position, be able to supply a fair amount of energy into the system. Perhaps more importantly though, it will also produce a small amount of energy on short, cloudy winter days. This is useful as the 200mA or so that the panel might produce on a cloudy day enables the

FEATURE

Figure 1
A budget-friendly 'Renogy' 100W solar panel

Figure 2
The Renogy 100W panel has plenty of mount holes built in – you can always drill and bolt into the frame if needed

Figure 3
The MPPT charge controller receiving 1.7A from the solar panel

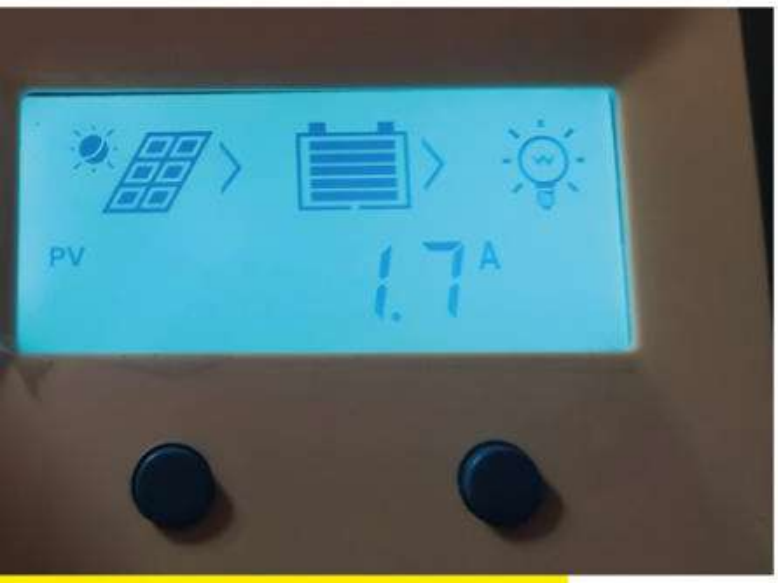
battery to maintain its charge, and whilst the charge controller may be drawing a small amount of current from the battery, it's still a net gain, and the battery is not discharging.

100W monocrystalline solar panels are a reasonably common product – I went with a Renogy panel because it had good reviews and was priced at £77 (**Figure 1**). It's pretty large at 1076mm by 509mm, but the aluminium frame that supports the unit is around 40mm deep. For its size, it's pretty lightweight and can easily be carried under one arm; if your arm is long enough to reach over it! Whilst I'm still deciding where to place it (the current option is on a small, secure frame on my shed roof), it's worth noting that the Renogy panel frame has lots of mount holes built in – there is nothing to stop you drilling and bolting to any area of the frame, avoiding the thin photovoltaic panel itself (**Figure 2**).

Battery choice is an interesting area to research as there are many different battery technologies to choose from. From traditional lead-acid batteries to lithium-ion, silicon gel electrolyte batteries, and more. Whilst larger and heavier lead-acid batteries are cheaper and have a long heritage of use in solar power systems, there are some drawbacks: the traditional 'flooded' lead-acid batteries, similar in appearance and structure to car batteries, require some ventilation and can only be used and stored upright. Sealed lead-acid batteries mitigate this issue, but if you want to build a system where the battery may be moved and rotated, you might want to look at gel-type batteries. Lithium-ion and gel-type batteries tend to be smaller and lighter than their lead-acid counterparts, so if you are planning a large storage system with lots of battery capacity, these would perhaps be preferable. The choice of battery was important in combination with deciding on a charge controller. At the budget end of charge controllers, you'll find they only support lead-acid batteries, so it's worth taking your time and looking over lots of options.

PWM OR MPPT?

A few years ago, the only real budget charge controller options were cheap PWM-style controllers. These are still available and offer an amazing amount of technology, often for less than £20. One thing to note with PWM-style charge controllers is that they are less efficient at power conversion than other technologies like MPPT (maximum power point tracking). PWM (pulse-width modulation), at best, is probably converting around 70–75% of the power from the panel. However, if you are interested in setting up the cheapest system you can to get going, then, at the prices they are, they are well worth considering. MPPT optimises the charge current supplied to the battery based



on the solar panel voltage, current, and the monitored state of the battery. In a nutshell, it chooses the best charge rate conversion at any given time. When you set up a system using MPPT, you can usually see that the solar panel side is generating a higher voltage than the battery needs and a certain number of amps, but monitoring the battery, you will see it is receiving a lower voltage but with a higher current. For example, in **Figure 3**, the solar panel side (labelled PV for photovoltaic) is generating 18V at 1.7A, whereas, in **Figure 4**, which was taken at the same time, the charge controller is DC-DC converting the 18V to give the battery 14.5V at 2.1A.

Having decided I wanted an MPPT charge controller, I looked at what was available. I'd read that my Renogy 100W panel was unlikely to produce more than 6A when in the UK, so I could have gone for a 10A-rated charge controller, which is fine for this panel with a safety margin; however,

Figure 5
MC4 connectors are relatively cheap and rugged and are straightforward to assemble and use

Figure 6
The numerous pieces that go together to create an MC4-type connector

I decided to spend a little more and buy a 20A-rated MPPT charge controller so that, if I decided to add an extra panel, I have the capacity to safely add it. The model I went for was a well-reviewed EPEVER Tracer 2210AN. Another advantage of this particular controller is that it can be reconfigured to work with either lead-acid batteries or lithium-ion, which again gives me options if I add or change the system in the future. Having made this decision, I paired this controller with a lead-acid battery designed to be used in solar power systems. The battery is designed to last three to eight years being periodically cycled (discharged and charged) and is rated at 110A hours at C100. The C ratings give an indication of how the battery would perform under certain loads; C100 means that if put under a constant 1.1A load, it would deliver that current for 100 hours. At C5, the battery is put under a 14A load and delivers it for five hours – a total of 70A hours. As our system is primarily going to be supporting less than an amp's worth of lighting with an occasional bit of around 5/6A to charge RC batteries plus other experiments, it seems a fine capacity for a starter system. The battery arrived well-packed and it had some travel plugs inserted to stop any leakage of the acid in transit. It's important that these are removed straight away – don't use the battery with them in. However, I kept the plugs and have taped them in a small bag to the side of the battery – this means that if I'm driving with the system disconnected, I could add the travel plugs back in if I felt it necessary.

"IT SEEMS A FINE CAPACITY FOR A STARTER SYSTEM"

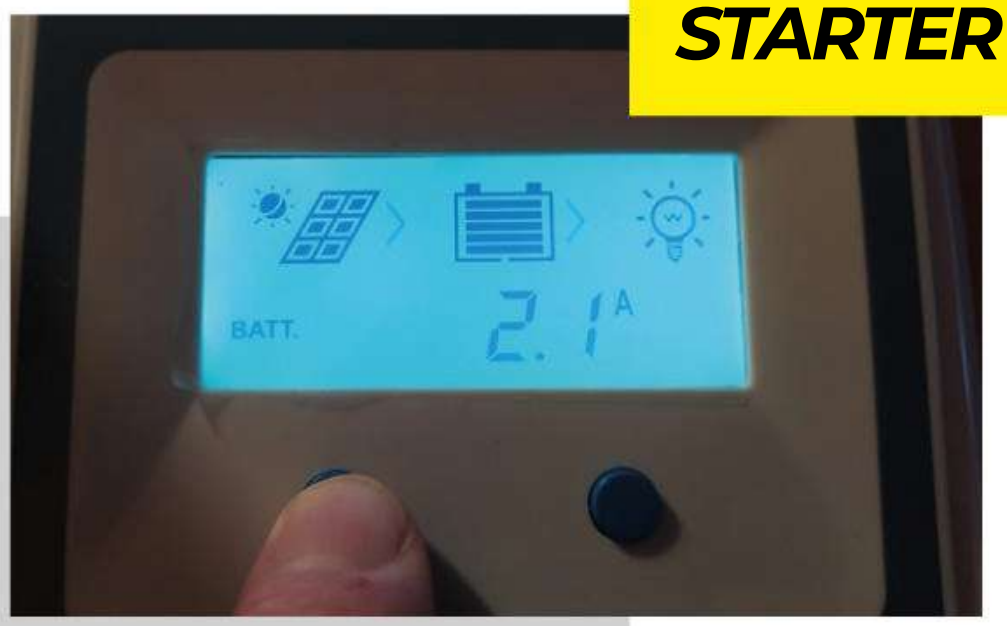


Figure 4
The MPPT charge controller having performed a DC-DC conversion is putting out 2.1A at 14.5V to the battery

FEATURE

GET CONNECTED

On arrival, the solar panel has a short set of cables around 60 cm in length marked as positive and negative and terminated in some IP67-rated connectors. These types of connectors are called MC4 – they're robust and weatherproof (**Figure 5**). I wanted to create some extension cables to run between the solar panel and the rest of the generator. I could have used slightly cheaper 10 AWG wire, but again, in case I wanted to extend the system, I bought two metres of 8 AWG flexible silicon wire, which is fine for up to 40 A. I don't plan to ever need that capacity, but it's good to oversize these wires for low resistance to minimise losses from the solar panel. MC4 connectors are easy to add: strip a small amount of cable and crimp a male or female connector onto the end and slide this into the case (**Figure 6**). Whilst they are designed to be left once connected, it's simple to connect and disconnect them as needed, simply squeeze in a couple of clips and they slide apart. I used a cheap crimp tool with a slot marked 5.5 mm, which worked well for the crimp assembly. The tool also arrived with a set of plastic spanners, which you can use to tighten up the MC4 casings to make them as watertight as possible. Whilst I'd recommend getting a crimp tool for the MC4 connectors, we've seen examples of people crimping them carefully with diagonal plier cutters.

The charge controller arrived preconfigured to work with a 12 V lead-acid battery, so I didn't need to change any settings. It has three different charge modes – these automatically kick in depending on the amount of available power from the panel, the battery state, and the load. Wiring up the charge controller is simple with an input from the panel, a battery connection, and a 'load' connector. The manual asks you to connect the battery first and then attach the solar panel. On the EPEVER unit, there is plenty of capacity to connect the 8 AWG-thick wires. At the battery end I found some battery clamps that had busbars built on, which allow you to make further connections direct to the battery (**Figure 7**). As a test, I used the battery clamps to connect and disconnect to power the system. Later I added a fused circuit-breaker, which adds both safety and a simple way to power down the battery side of the system. With the battery connected, the unit springs to life and then you can connect the panel. The interface for this is simple: you press the select button and it scrolls through the screens that give you information on the solar panel, the battery, and the load. To attach something you want to supply power to, it's common for people to use the battery connectors connecting into the system with a suitably rated fuse. Additionally, the EPEVER charge controllers also have a 'load' output – this isn't useful for my intended use, but the 'load' connection can be set up in different modes and allows you to control the connected device, which is probably aimed at lighting. For example, you can set the load channel to only be powered for a certain amount of time after the panel has stopped charging – very useful if you

Figure 7
These budget battery clamp connectors have integrated busbars, so you can attach other wiring

Figure 8
The complete case before painting and system fitted, plenty of ventilation and room to stow cables and accessories

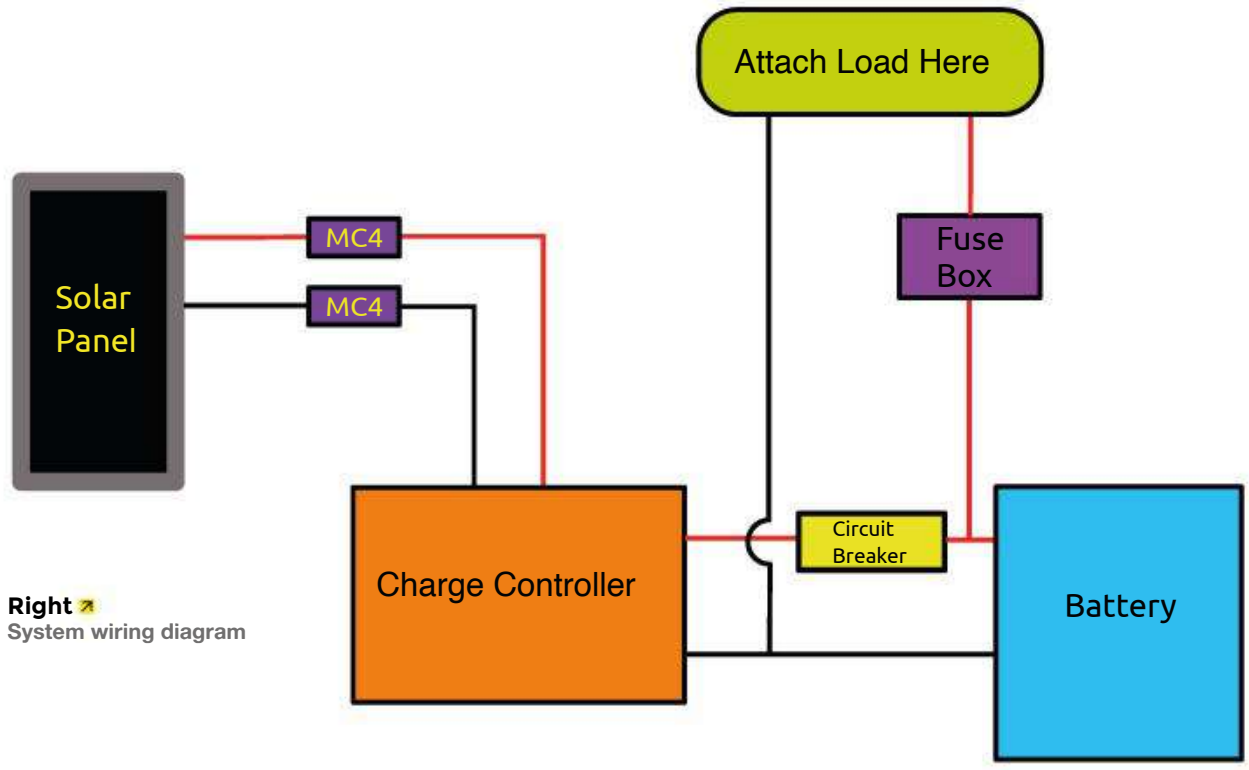
Figure 9
These 20 A in-line circuit-breakers, originally designed for high-power car audio, were a great find. They're perfect for isolating different parts of the system

Figure 10
Adding a fuse-box on the positive output means we can add suitably rated in-line fuses for the attached loads

want to have automated lighting that runs for a few hours after nightfall. You can also configure the load channel to be always on or always off. Of course, using the load channel also means that the load can be switched depending on the battery condition. This means, for example, if lighting is attached and is accidentally left switched on when the battery reaches the safe discharge threshold, it will power down the load channel output. The charge controller is backlit, but only for a period after a button is pressed. As an experiment, I used a multimeter, and even when the back-light was on and I was switching between screens, I read a current consumption of only 26 mA, which is excellent. The charge unit also came with a thermistor, which you can attach directly to a small port on the unit, or you can run a wire from this port to place the temperature sensor near the battery. This again protects the battery, and if the battery reports a high temperature, it will shut down the charging system.

With everything in place and tested, I began to make a wooden case to hold everything together (**Figure 8**). I wanted to keep things simple and strong and also well-ventilated. The charge system and battery need to be ventilated, and the charge controller states it requires a minimum of 15 cm in each direction when mounted to a wall. I decided to mount the charge controller to the top of the case so that air can freely circulate around it. I made the lower deck long so that it has enough room to stow cables and accessories alongside the battery. The battery is quite heavy, so it's always going to be a two-handed lift and carry, so I wasn't too concerned about adding weight with the timber.

I made the frame from 9 mm plywood, using 20 mm square pine to make internal parts to screw into. The base, back, and sides are all screwed and glued to make it solid. The top/shelf is screwed into position but not glued, this means I can remove the top if I need to access the battery, make changes, or reroute cables. As well as the top shelf having the charge controller mounted to it, there are a few other additions. For safety and ease of powering down the system, I added a 20 A-rated



Right
System wiring diagram

Below
Charging LiPo using a 12V RC charge system will be really handy at events

circuit-breaker. These circuit-breakers are designed for high-power car audio but are often used in solar power systems (**Figure 9**). You can quickly switch the breaker between the open and closed position, which is mounted on the positive cable between the battery and the controller. To disconnect the solar panel side of the system, I unclip one of the MC4 connectors, but I think it would be a good idea to add another of the breaker switches on the solar panel lines, meaning I can isolate all parts of the system. I've also added a small automotive fuse-box to the top shelf (**Figure 10**). This is used to connect loads to the battery safely with different in-line fuses. We wouldn't use anything on this system that would require over 10 A currently, but we can swap in lower or higher value car-type fuses as needed. It's also possible to buy single in-line automotive fuse holders rated up to 10 A – these are useful as they can be wired in-line to devices that we might use less frequently.

With everything laid out correctly in the case, we removed everything and painted the case to seal the wood, as it will spend most of its time in a damp shed! To finish off the case, I added a couple of cast iron handles to the sides, which make it much easier to lift.

This project has already started to earn its keep as it's running the low-power 12 V lighting I had in my shed. I've also been using it to charge LiPo batteries for my RC and robotics projects. It gets you thinking about what you could run off a 12 V system. I'm looking at converting one of my smaller 3D printers to work off the power of the sun!

HackSpace magazine meets...

Jorvon Moss (aka Odd Jayy)

A creator, 3D designer, and self-taught robot maker

Jorvon Moss is an artist by training. From his workshop in LA, he makes robots – not to do anything like send people into space or fight crime or any of the stereotypical things that comic book fans are supposed to want to do, but just for fun. He's also an obsessive documenter, so if you've ever wanted to know how to build (or how not to build) anything with embedded AI, or servos, or face recognition, he's a good person to ask. We spoke to him in the early hours one morning, to ask him what he's been up to recently, how to get into building with artificial intelligence, and how he approaches the creative process. →

Right
Jorvon started out frying batteries for fun; now he's a full-time maker (also for fun)

Jorvon Moss (aka Odd Jayy)

INTERVIEW

HackSpace So Jorvon, you're a full-time maker now. How did that come about?

Jorvon Moss I was working full-time at an access control company, doing that full-time and making things in my free time. But then my company got bought by Motorola. And since that happened, they bought everyone stock, so that gave me extra money. I stayed for like another month, just because I thought I'd try and see if I can get used to this, but I honestly couldn't get used to it.

I had a very serious back-and-forth with my boss and he told that it just wasn't for me. So, that same day, I put in my two-week notice. I asked the internet brain for help, because I was like, hey, I want to be a full-time maker now because I can't do nine-to-five anymore. And the people at Adafruit actually were the first ones to reach out and be like, hey, you sound awesome for this position. They reached out to Digi-Key, and I started working for Digi-Key for a while as a freelancer. Their technical content creator job came up and they offered it to me, so now I make for a living (**hsmag.cc/JorvonMossDigikey**).

During the lockdowns, I've had more time to work on stuff and I was able to develop my skills better.

I got a lot more into wearables over the years, but I still focus mainly on robots. That's my thing. I'm currently working on one right now, which I literally took apart last night again.

It's not done yet – I have to add an extra jaw point to give it a mouth. I'm hoping to have this done soon so I can get pictures of it and document it properly. Most of it has been, like, upgrading software I use.

I've been practising with AI. I've mastered facial recognition, now I'm moving on to other stuff.

HS How did you get started on the road to AI and robotics?

JM I started with simple small builds, like everyone else does. I made this little cup thing, I designed and 3D-printed it myself. And then of course, I saw Alex Glow's owl, which inspired me to build my spider. But then I wanted more. So I kept going. I just kept building and building and building and building and building. And now I do this, but I started with just a 3D printer and CAD in my room. I'm all self-taught when it comes to making all these different forms.

HS How does someone with no mathematics or computer science background get into facial recognition with AI? Because that sounds really hard.

JM I was trying to teach myself years ago before I got into it, because at one point I felt like I had mastered using certain boards like Arduino. I mastered basic Raspberry Pi stuff so I could make servos move in a way that I thought was useful, and I wanted to make my robots do more, and AI was the next step into that.

> "I think it took me about four months to finally get the facial recognition software working on one of my robots"

So again, I used the power of the internet, and found tutorials on how to do AI, and I've just been slowly picking it up one by one. And then eventually, I started testing my AI systems, just to see if I can actually get facial recognition to work.

I got help from friends, of course, colleagues who are able to put input here in there and be like, 'Oh, this line of code is wrong. You did this wrong, you should have done this'. And I'm like, 'Oh, well, thank you. I'll try this.'

It was great. Honestly. It took a while – I think it took me about four months to finally get the facial recognition software working on one of my robots. But now I keep that stuff together; that way, if I ever want a robot with facial recognition working efficiently again, I can easily just set it up. I want to keep moving forward, I want to keep adding stuff.

So now I know facial recognition, I want to get into other things, like voice recognition, making smarter robots that can be worn like compatible type of things. I actually built a robot backpack, which is easier than it sounds.

I plan to do a little bit more travelling this year so I built this bag. That way I can have a very artistic and custom way to carry my robot. And also something I learned from a design point of view, having a robot strapped to your shoulder causes problems. It looks cool from far away, but when you get close, you always have the leaning problem with the robot leaning forward, or off to the side a bit. So I built this backpack to have an extra little stand on it. So my robot actually stands on the bag more than on my shoulder, which actually helps visually, because it's looking over my shoulder. It gives it a vantage point to look around.

HS So it can look around and recognise and react to faces? I guess that'll get some reactions.

JM Yeah, I'm working on two different versions of this robot, one with AI in it that's doing facial recognition. But the current one, I'm just controlling with my phone, because I don't want certain people to get scared.

I took a prototype of my facial recognition bot to SiliCon last year, and it got some love, but it also got some bad reactions too.

And also, I found out how flimsy facial recognition really is. Because the thing is, the robot wouldn't notice you if you were wearing a mask. So if you had your mask on, you're invisible to my robot. It couldn't tell you had a face, but as soon as you pulled your mask down for a second: oh look, there's a human. →

Above ↗
The best way to learn is to have a go – trial and error is better then theoretical learning

Above
This is Helen, a robotic display built into a backpack, as seen in a Los Angeles near you

HS We can forgive that in a robot though, right? Less so in a self-driving car.

JM Luckily, this doesn't do anything too crazy or important, it's just designed to inspire. That's one of the biggest parts of my brand, literally just making for fun, and I really want to continue to, like, push that ideal. I really don't make with actual purpose like most people do.

The thing in the big engineering spaces is always like 'making to make the world a better place', or making to do this or that. I like to think that I want to just make because making is fun. You shouldn't have to have a mission to save the world with your project. Sometimes it's just nice to build something. Make for fun. Make for kicks and giggles. Why not? Go ahead and build what you want. Who's going to stop you?

HS What sort of reaction do you get when you take your companion bots out into the world to show the public?

JM I had a talk in San Francisco last year, when I went and showed my robots to kids, and kids got to see the facial recognition, see how they move around and stuff like that. Kids love them. A little bit too much, because I had a few kids try to steal them.

I really want to do more talks in general, I am 110% here to do that. But I think that's one of the main reasons I want to make this one cell-phone-controlled; that way I could actually interact with people a little bit more. Because with facial recognition, AI – at least this is what I've learned – so much has to happen for facial recognition to work correctly, like perfect lighting for example. Sometimes when you're out and about [with] the sun behind you, my robot can't see you. It's very finicky.

HS What sort of hardware do you use to get facial recognition into a little box that sits on your shoulder?

JM I use two different things, depending on the size of the project. I use the Raspberry Pi with the camera attached to it, and then do the whole open-source facial thing to attach it to a simple servo for pan and tilt, and power the two components separately; or I use the OpenMV (**openmv.io**) and a Pololu Maestro servo controller. I have those two boards talking to each other, and then that can also do facial recognition very well.

If you really want to get into easy AI stuff, I would definitely recommend the OpenMV cam, especially for the fact that it comes with some stuff pre-loaded already. So you can plug it up to your computer and instantly start seeing some example code for facial recognition, identification, and stuff like that. It's a perfect starter board.

> "Turns out that everyone around me was wearing a mask – it can't see me when I'm wearing a mask"

HS I flew back into the UK from abroad last November, and the facial recognition passport scanner was absolutely stumped by the fact that I've grown a beard and was wearing glasses.

JM You have to see facial recognition as being in its infant stage, that's how I see it. Like it's, it's still a baby. It's like a toddler who finally hits three and finally has a decent type of brain to figure out what stuff is. It's still learning. It's going to be asking 'What is this? What is that?' Trust me, when I first took my first facial recognition robot out, it didn't recognise any people at all, and I thought I'd done something wrong. Turns out that everyone around me was wearing a mask – it can't see me when I'm wearing a mask.

It's those little things. I just assumed it was looking for facial shape. No: it's looking at everything. Eyes, nose, mouth... it's looking for all those things to be present, or it won't recognise an object as technically a face. Unless, that is, you teach it yourself to look for other things, like eyes, or the presence of a person's body, or something like that.

HS If you're mostly self-taught, does that mean that you've spent a lot of time hanging around makerspaces?

JM Not particularly. I do my best learning by myself. I'm that nerd kid who grabs a whole bunch of books and stuff and then just, like, sits at a friend's computer and writes things down and starts testing things like crazy. Makerspaces are great resources for people who don't have the equipment to do the things they need to do. And luckily enough, I have the equipment. But my main problem is my focus – I feel like I have ADHD brain. I focus better at home, in front of my desk, because every time I go into a makerspace, I end up doing more talking and hanging out than actual work.

When I'm studying, I try to make a single goal per day. So it'll be something like 'Alright, let's get the camera set up today.' Once the camera is set up and it can see me, I'll look up what I have to do next, and the next day I try to do that.

I build upon trial and error on a daily basis. Every single day I'm trying to build this, trying to build that, and some days are good; some days work perfectly. And some days, things have to be redesigned and reworked. This is the fifth time I've rebuilt the robot I'm working on currently. I literally took it apart last night because I didn't like the way it was moving. I started to add some spare parts to it, reattaching them, and it worked. I just have to make it a little prettier.

HS Speaking of things looking pretty, how do you finish a project off? How do decide that something is finished?

Above
Jayy's personal robots are never finished; they're pulled apart and improved constantly

JM Fun fact: I don't decide when something is finished. The internet does! I personally believe that nothing I do is perfect, nothing I've done is good enough in my opinion. So, what I end up doing most of the time is going back and rebuilding it; at least getting it to a point where it's functional and it looks good. And then I'll post it and, depending on how badly the internet wants it, [that] determines whether I develop it or not.

But to me, nothing is ever really done. It's always based on deadlines; if a deadline says something has to be done by a certain date, I'll take features out of a build to get it done. Because personally, I don't think any of my work is actually to the point where I'm happy with it. Perfection is an illusion.

When I'm working for a company, I publish work more often. But for my personal work, it all depends on the internet, the community. For example, the goggles that I made were really popular. Once I made them, I made another pair to make sure that everything worked, and then I published it, because the internet wanted it. But to me, it's still not done. Like I could add more moving parts to it and add this cool thing and this cool thing... That's why I just call them prototypes. That way, if anything doesn't work, it's OK.

HS You've taken a lot of pressure off yourself by using that word.

JM Exactly. People have a lot of preconceptions of what a perfect project is thanks to Apple et al. And then, of course, you have movies – all of these tech characters like to build things in like five minutes without any real parts. They somehow glue imaginary material together and make a thing.

I'm by myself; I don't have a team of people who are building these things for me; I got to do the best I can. And, usually, I just think of one thing that I want to be the main functionality and work from there.

HS Am I right in thinking that you're following the convention of naming your builds after mythological creatures? You've got robots called Odin, Prometheus, Asi, Helen...

JM No, Helen was based off of Helen [Leigh] from Crowd Supply because they bought me a very expensive part. And I appreciated it so much that I named a robot after them. That's actually how I got my first OpenMV Cam, because it was really expensive and at the time I couldn't afford it.

But I posted on Twitter (I love Twitter – it's like a giant internet brain) like 'Hey, I'm looking for some help. Does anyone have an extra one of these lying around?', and Helen sorted me out with one.

HS Let's keep that quiet or everyone will want one. It sounds like the internet is pretty useful for you as a giver of second opinions, not just for finding facts.

JM I feel like currently, the internet is as complicated as a human. And what I mean by that is that there are different layers of a person, like, no person is perfectly good and no person is perfectly bad. I feel like the internet's, kind of, becoming slowly human that way.

I love the internet. I love the fact that I can, like, entertain myself for hours, watch movies, build things, reach out to people that I could not talk to in person. I love these things, but at the same time, it's also a place for people to escape from reality. It's 100% a tool for me that I try to use for good.

HS What do you have planned for your next build?

JM I got a Raspberry Pi 4 recently, which I'm working with to build my own AI, to put into an R2-D2-sized personal robot. I want it to be more intelligent and more useful than my other things, so it's going to take a while. I want to have this really big robot to walk or roll beside me, and have it [be] intelligent [enough] to look up when people are talking to it, so it looks like it's following the conversation.

I've talked to a few of my programming friends and we discussed the best method of teaching, which is probably going to be to treat it like a baby – showing it the items and getting it to repeat what those items are back to you. By necessity, that's going to be a slow process. So, it's gonna take a while. That's fine. I mean, we all like babies. They're small. They're cute – it's just that mine will have an off-switch.

Above
Humans, fear not: artificial intelligence isn't that intelligent. Your apex predator position is safe for now

FEATURE

SOIL

Mud, mud, glorious mud, is the foundation for many fantastic makes, finds Rosie Hattersley

Rosie Hattersley

@RosieHattersley

Rosie Hattersley writes tech, craft, and life hacks and tweets **@RosieHattersley**.

Soil has been around for 450 million years and has been used for farming and construction for many centuries. Although soil ultimately results from the weathering of rocks, it didn't exist as we knew it until plants began growing on land rather than only in the sea. This helped stabilise the atmosphere, removing carbon dioxide and helping cool the Earth, making it suitable for life. Soil has been integral to our own and other creatures' lives ever since. As well as being a medium for growing crops, soil is a natural filter, cleansing water of pollutants, and making it safe for drinking. As with water and minerals, however, soil is a finite resource, as artist Kirsten Kurtz's video interview (see page 60) reminds us.

> "Excavated soil from construction sites is the biggest source of waste in Europe, but 80% of it could usefully be reused"

Frustratingly, excavated soil from construction sites is the biggest source of waste in Europe, but 80% of it is uncontaminated and could usefully be reused elsewhere. An aftermarket aimed at DIY homebuilders and landscape gardeners seems like a fairly obvious means of saving it from landfill.

Most permanent structures are, of course, made from clay-soil, aka bricks, and you could certainly make use of heavy clay-soil to make your own. Brick formers are readily available from trade and DIY stores. Mud huts may seem like a primitive use of soil as a housebuilding medium, but there's renewed interest in constructing dwellings without resorting to pouring concrete in the form of packed earth or cob-houses.

Different types of soil, of course, have different uses, not only lending themselves to growing particular crops, but also for varying forms of construction. If you're not fully sure which type you have and are keen to start gardening, there's a helpful guide to soil types across the UK that you can consult by entering your town: **hsmag.cc/Soilscapes**. The other option is to get a pH monitor from a garden centre plus an ericaceous mix or other balancing compost to make your beds suitable for the blooms you hope to grow. Some vegetables, such as Brussels sprouts, need the firmness of a clay-soil to thrive.

Pottery clays are sticky and heavy, contrasting with soils rich in lime, sand, or loam that help adjust the pH balance but also break up clumps of mud and aerate the soil so plants can grow more efficiently. Spent compost and less than nutrient-rich soil will, however, do a very good turn in the base of a raised bed or large plant pot, saving you shelling out for additional sacks for the new season's crop.

BRICK IT!

Ecologists keen to make good use of soil are already using it in bottle walls (as per HackSpace issue 49) as well as making their own bricks to keep construction costs down. Sourcing clay for brickmaking from a nearby riverbed might be an option, suggests Steve Nubie. He "learned the hard way" that his property was "blessed" with clay every time he tried to plant a tree or bush, discovering hard clay six-to-twelve feet beneath the top-soil. He used bricks from his own back garden to build a wood-fired pizza oven, spreading his reclaimed garden clay over sheets of plywood and kneading it with his feet to make it malleable. Once he'd got a pliable, elastic consistency, he mixed four parts clay with one part sand. After priming the 8×2×2.5-inch brick formers with charcoal dust, he poured in his clay-sand mix, wired off the tops to level them and left the bricks to dry slightly on a plywood sheet that he'd greased with vegetable oil. Twenty minutes later, he removed the formers and let them dry out for several days. A three-foot bonfire built around stacks of eight bricks was used to fire them. Cooling takes up to a week, after which they can be used to construct walls, pizza ovens, and other small structures. →

Project Maker
Steve Nubie

Project Link
hsmag.cc/MakeYourOwnBricks

Left ◆
Pizza tastes better when it's cooked in a homemade oven

FEATURE

SOIL PAINTING

Project Maker
Kirsten Kurtz

Project Link
hsmag.cc/SoilPainting

Kirsten Kurtz combines her role as manager of the Soil Health Lab at Cornell University with a passion for painting with soil. She loves the shades, properties, and textures of different soils, and uses them to incredible effect in her *Painting With Earth* portrait series. She says using soil painting as a medium for communication has been incredibly successful and as a means of getting people to think about soil as a natural resource that deserves to be protected and cherished.

Right
Celebrating the diversity of soil

EXTRACT YOUR OWN POTTERY CLAY

If *The Great Pottery Throw Down* has got you thinking about creative crafts, you might have pondered whether you can make good use of the clay-soil in your own back garden. Most advice suggests it's better to buy clay than to take a chance on the composition and suitability of the clay-soil you have to hand. But if your needs are modest, in theory, you can make use of back garden soil. Justin's Makery advises you to collect clods of dry soil that is largely free of weeds, roots, and evidence of critters living in it, add some water, break it all up with a spade and crumble the remnants. You might not discover much clay, but almost any land with compacted soil should yield some. Add some more water, then pour the liquid mud/clay mix through a colander and into a second bucket. Any rocks and sand should get left behind in the original bucket. Let the bucket sit for a day or two to clarify into water and clay, pour off excess water and then upend the rest into an old pillowcase stretched over the top of the bucket. Allow it to dry out until it's a "fairly manageable damp lump". Roll and pat it in a cloth to dry it some more, then wrap in plastic until you're ready to use it. Justin, a regular potter, reports it is "very soft with a kind of dry skin to it". Nonetheless, he managed to use backyard soil to create his own crockery for free. For noticeably better results, try a 50:50 mix with regular studio clay, he advises.

Project Maker
Justin's Makery

Project Link
justinsmakery.com

Left
People have been making their own clay for thousands of years

Below
What would it be like to live inside a clay pot?

CASA TERRACOTA

Built entirely from soil, Casa Terracota may well be the largest piece of pottery in the world.

Known as 'the Flintstone House' to locals near Casa Terracota in Boyacá, Columbia, architect and maker Octavio Mendoza Morales built the 500 sq m sustainable dwelling as an elaborate proof of concept after one of his nieces asked whether the clay pot he'd presented to her could theoretically be used as material for a habitable house. Bricsys (**hsmag.cc/ClayPotHouse**) reports that construction on the resulting home lasted 17 years and involved clay-soil sections being baked for a month before being left to cool for a further week.

Keen to keep the house as eco-friendly as possible, Octavio went on to use local clay to make the dishes, along with upcycling metal and glass for some of Casa Terracota's fixtures and fittings. He even used clay for the bed base and other furniture items, with mosaic tiles and whimsical chimneys giving the building a Gaudí-esque feel. He told NBC (**hsmag.cc/CasaTerracota**) that his goal was to demonstrate how soil can be transformed into habitable architecture by simply using the natural resources at hand, making a point of using no cement or steel in its construction.

Project Maker
Octavio Mendoza

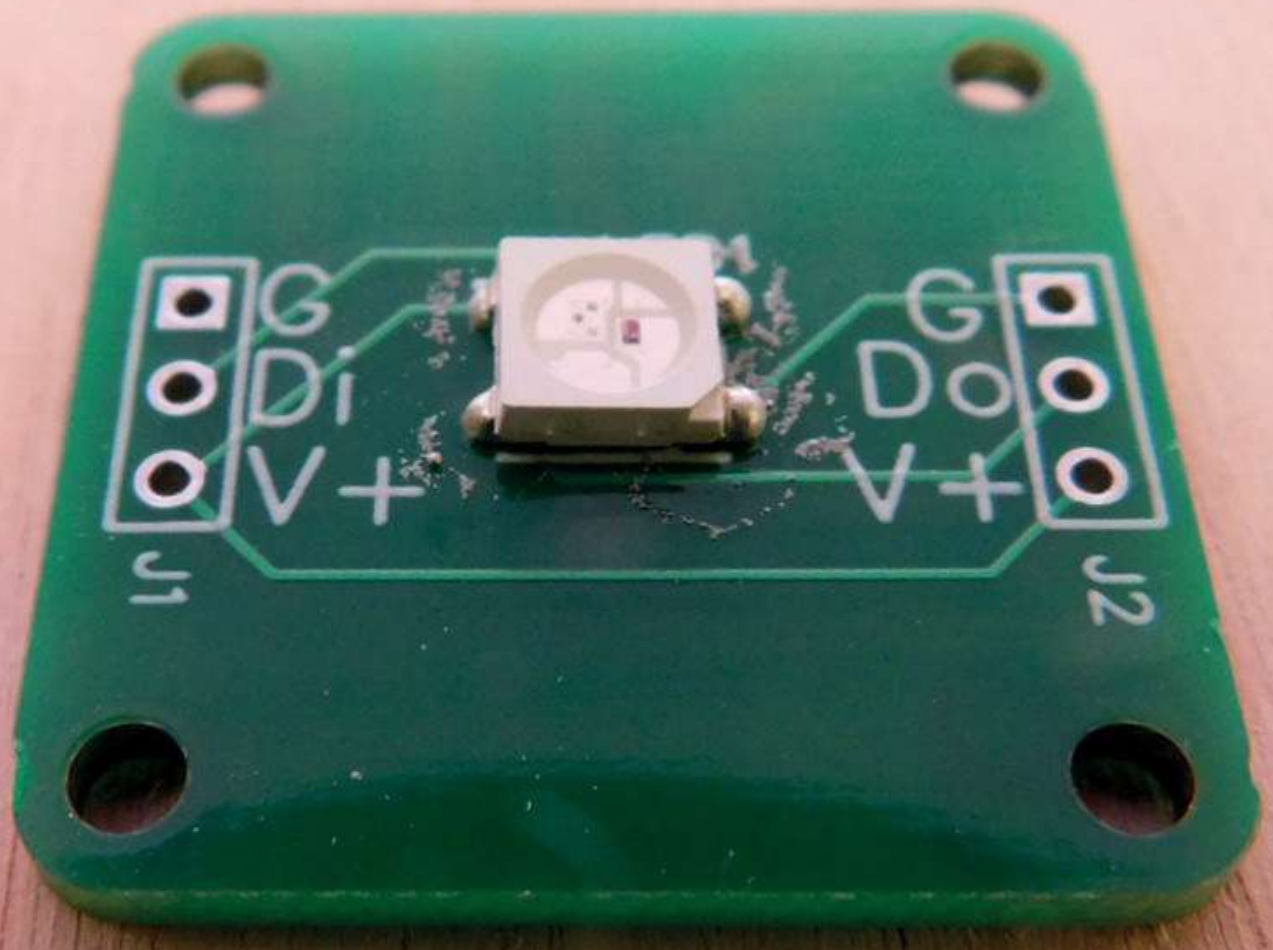
Project Link
casaterracota.com

FEATURE

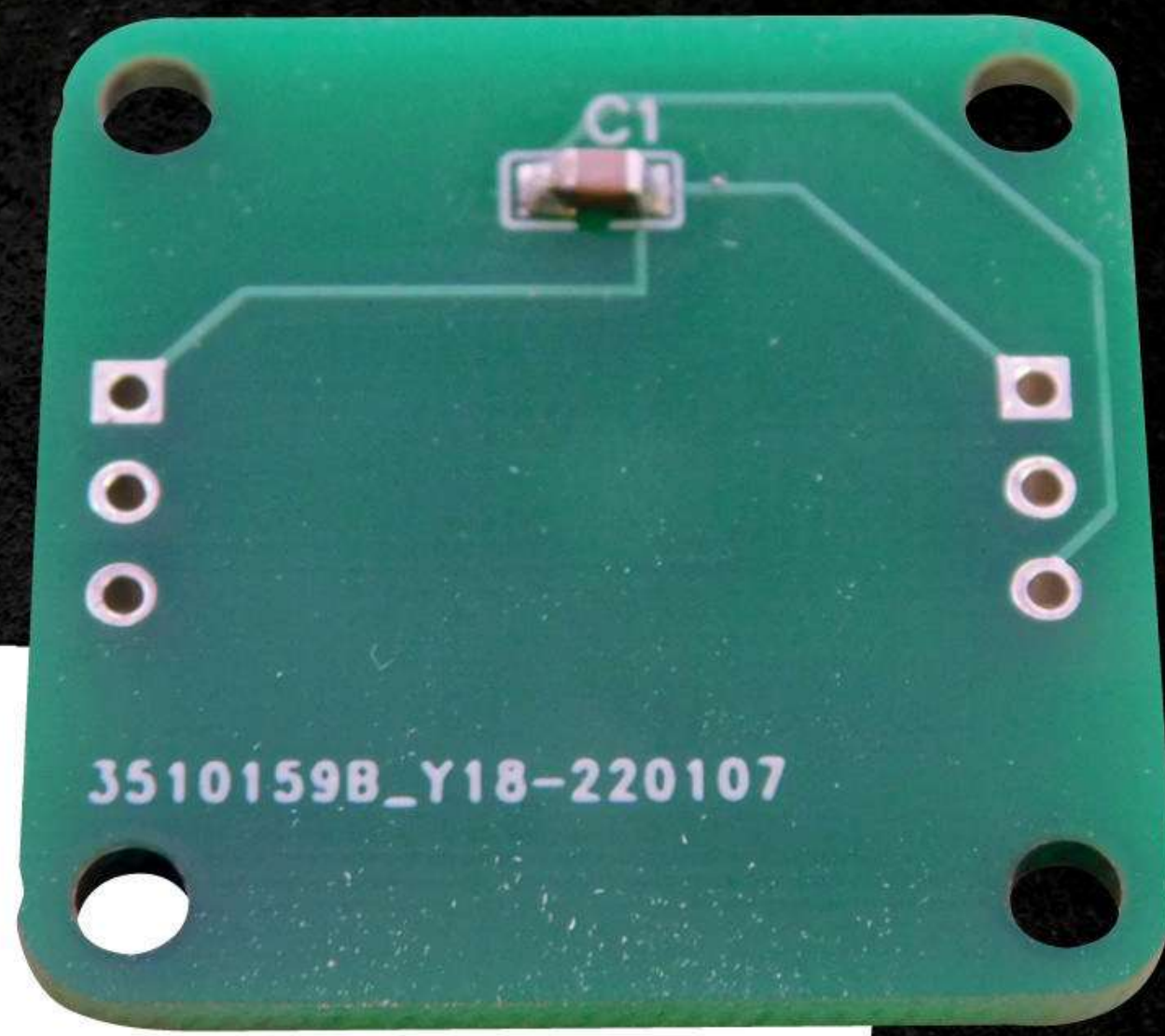
IN THE WORKSHOP: A dot of light

By **Ben Everard**

Yet another WS2812B board



Right
It's a WS2812B breakout that is designed to fit into other projects



Above
The capacitor is on the back so it can mount flush against other things

There's a huge range of options for incorporating WS2812B-addressable LEDs in your projects. You can buy strips, matrices, individual LEDs, breakouts, and more. Does the world really need another one? Well, maybe.

In the last few months, we've been experimenting with shining light through PCBs to create different effects. For simple LEDs, you can just mount them upside down – a decent blob of solder will bridge the gap, and it's not too hard to get the technique to do this working. But what about WS2812B LEDs with four pads and a different footprint? For these, we found we needed a bit more of a connection, and it wasn't easy to just mount a single LED upside down.

There are a few options for this – gluing a section of LED strip down, for example – but none of them really seemed like a good idea to us. Despite all the various WS2812B modules available, none seemed particularly suitable for mounting on PCBs. For that matter, none of them are particularly good for mounting on wood or 3D-printed parts. For our breakout boards, we had a few requirements:

- Nothing on the front other than the LED, so it can be mounted in a cut-out
- Mounting holes so that it can be bolted onto something
- Convenient placement of headers to make chaining easy

These don't seem particularly onerous, so we're really surprised that no module on the market seems to fulfil them, but as far as we can see, they don't. We fired up EasyEDA, and set about making this module a reality.

The schematic is really simple – it's just three pins for voltage, ground, and data-in on one side, three pins for voltage, ground, and data-out on the other and, in-between, a WS2812B and smoothing capacitor between voltage and ground. The PCB is similarly simple – the LED is on one side, the capacitor on the other, and there are mounting holes for M2 screws in each corner.

We had this knocked up and sent to be manufactured in about an hour but, just afterwards, we were doing a bit of research and came across an article on vintage audio PCBs, and noticed that the hand-drawn traces on old PCBs looked so much better than ours. Could we make something a little more easy on the eye?

We found we needed a bit more of a connection

We played around with EasyEDA's tools. While you can't create arbitrary bends and curved traces, you can use the circle and arc tools in the copper layer. We wondered what would happen if we routed the entire board using just these tools. Since our board has only a few components and a simple →

Above
Different fonts and cutting depths give different effects

schematic, it proved pretty easy to get it routed like this.

There's no point in spending time routing your board with strange traces if you're not going to be able to see them, so we wanted to show them off. We considered the option of exposing the copper and not having a solder mask, but this seemed a bit extreme, so instead we opted for OSH Park's 'After Dark' service, which uses black boards with transparent solder mask and gold coating. When it works well, it looks glorious. However, there are delays with their USPS postage at the moment, and we couldn't really justify the costs of fast shipping for some twirly LEDs, so we're still waiting to see what they look like in real life. Hopefully, we'll be able to bring you news on that next month.

When it works well, it looks glorious

We do, however, have the boring-trace version back from manufacture, and while they don't look as good as the alternative version, they do work, and this is one set of PCBs that we wanted to be functional.

Now we've got these back and they're functional, we'll start pressing ahead with our back-lit PCBs that we've been experimenting with.

ENGRAVING

A CNC PCB mill is a useful bit of kit for speeding up your PCB design-test-fix cycle. They're increasingly inexpensive, and not too hard to use. While PCB manufacturing has become cheap, fast postage has not, so an error in a design can mean a long wait for the next revision. These CNC mills aren't only useful for PCB milling. You can use basically the same hardware for milling other materials. We're still getting to grips with the one in our workshop, so we decided to test out our hardware and software toolchain for engraving brass. This is basically the same as milling a PCB only, rather than creating traces, we want to create words, and obviously the material is a bit different.

First up, we're going to use Inkscape, our go-to vector image editor. It's free and available on almost

all computing platforms. You can use the text tools to write out any message you want. We could be more elaborate, but for a simple test, I just etched 'HackSpace magazine'.

I played around with a couple of different fonts to see how they would look in brass. I wanted to try out some single-line fonts that I've used on the plotter before. I added these using the Hershey Text extension created by Evil Mad Scientist Laboratories for their plotters but, unfortunately, these didn't work well with FlatCAM. I'm not too sure why.

The tool I'll use to convert the SVG into a toolpath (FlatCAM) doesn't like the paths that Inkscape creates. The problem is that it creates a single path for each letter, and some letters have holes in the middle which require a second stroke to draw. Inkscape makes this section of the path invisible, but FlatCAM doesn't understand this. We need to break each letter apart. If you select the Edit Paths By Node tool (it's an arrow pointing to a line with squares on), and hover over some text, you should see a red line showing the path. If there are two red lines, then we need to break them apart. First, click on the red line to select that path, then go to Path > Break Apart. Now, if you click away to deselect the letter and then hover again, you should see that only one of the two lines goes red at any one time. You need to do this for each letter. Once this is done, you can save the image as an SVG, and it's time to start working on the tool path.

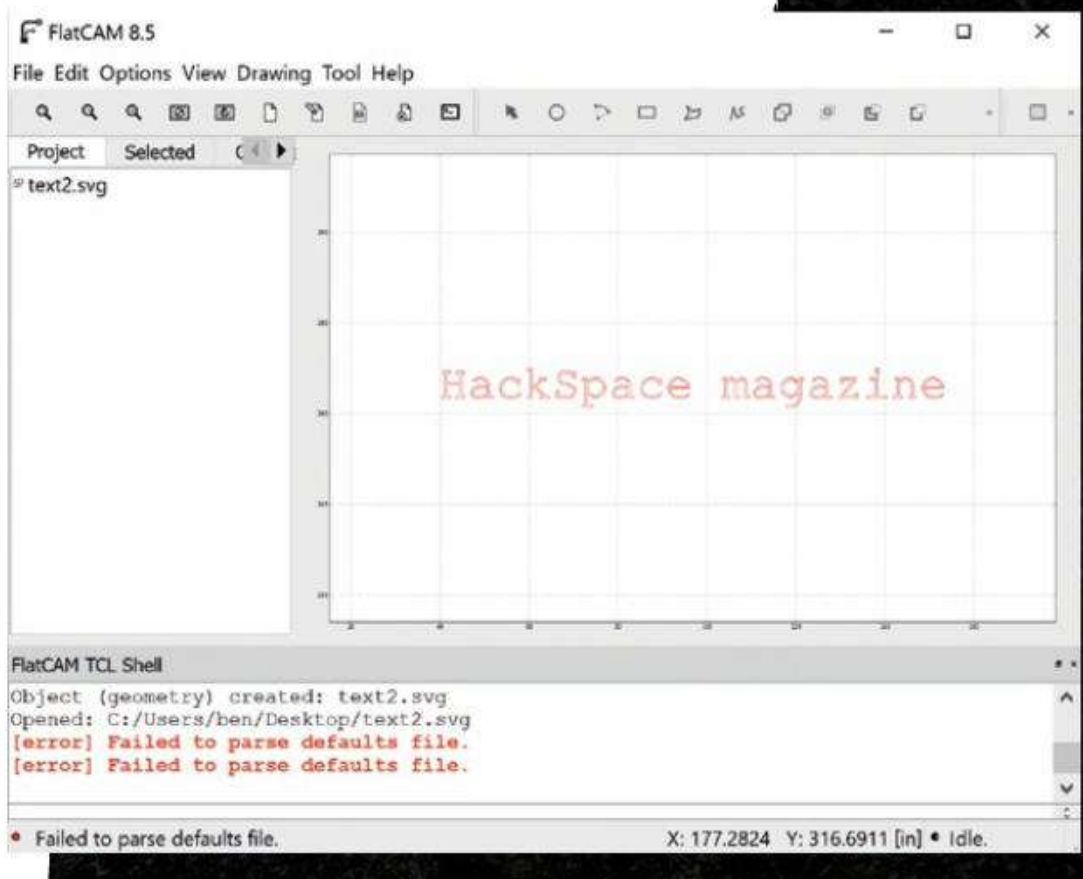
FlatCAM may not be the obvious tool for the job (or it may be – to be honest, I'm pretty new to the world of CNCs). I chose it because it's the tool I've been using to create toolpaths for PCBs, and I often find it useful to push new tools in odd directions when learning them. It helps me to learn about their limits, and the finer points of their use, more easily than when just using them in the 'normal' way.

Working with SVGs in FlatCAM is a little different to working with Gerber (PCB) files because, with an SVG, you already have a 'Geometry'. This means that the path in the SVG is the path that the CNC will cut. With Gerbers, you have copper areas that you then have to mill around.

To generate a toolpath from an SVG, first import the SVG using File > Import SVG. This will create a geometry object that you can select in the project tab. Then, in the Selected tab, you can use the scale and offset options to get the text in the right size and place. The easiest way of doing this is by putting the mouse cursor at the bottom left of the first letter (or wherever you want the origin to be), then, in the bottom right of the screen, you can see the XY coordinates of the cursor. If you make these negative, this will give you the vector to move the text to the origin.

Now it's time to generate the toolpath. This takes the geometry and adds the depth of each pass and the number of passes, and creates G-code that the machine can understand. In the CNC control software (we're using bCNC), you can override the feed rate and spindle speed, so we played around with these.

I'm still quite inexperienced with the CNC, so might not have the feed rate, depth, or spindle speed figured out. The cuts were pretty messy, so I cleaned them up with sandpaper. The results aren't perfect, but they definitely show promise. I want to get to a point where I can make little brass labels for builds, or perhaps even faceplates. I'll also try aluminium to see how that gets on. There's a lot to learn here, but I feel like it'll open up a whole lot of options for my making. ◻



Below
FlatCAM can import SVGs, but they can have rendering issues

Raspberry Pi PRESS

The MagPi | HackSpace TECHNOLOGY IN YOUR HANDS | CUSTOM PC

3 ISSUES FOR £10

+

FREE BOOK

hsmag.cc/hsbook

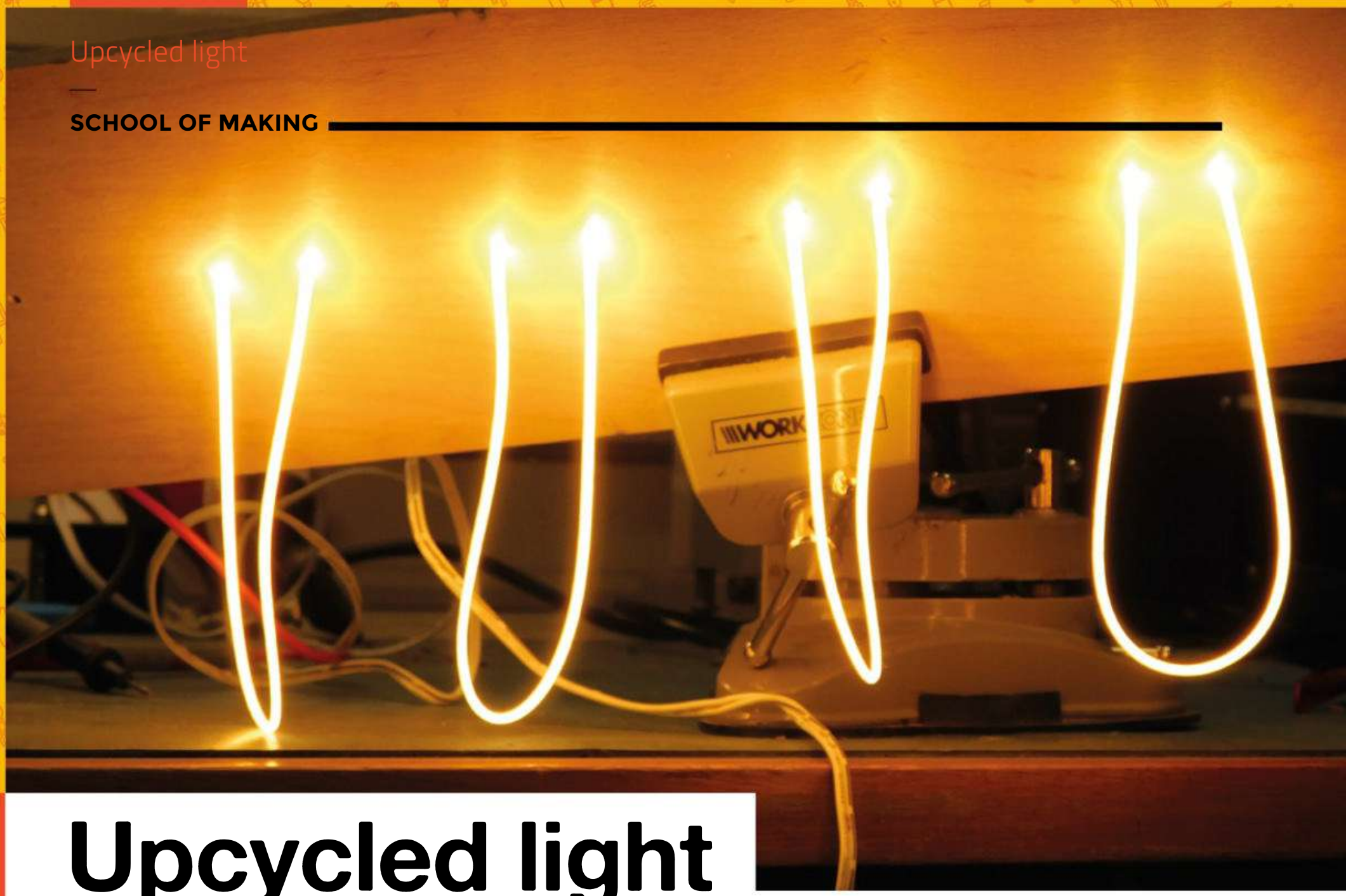
Subscribe to The MagPi, HackSpace magazine, or Custom PC. Your first three issues for £10, then our great value rolling subscription afterwards. Includes a free voucher for one of five fantastic books at store.rpipress.cc/collections/latest-bookazines
UK only. Free delivery on everything.

HackSpace
TECHNOLOGY IN YOUR HANDS

FORGE

HACK | MAKE | BUILD | CREATE

Improve your skills, learn something new, or just have fun tinkering – we hope you enjoy these hand-picked projects



Upcycled light

Adding some flexible LED strip to bits from the spares bin

Ben Everard

@ben_everard

Ben's house is slowly being taken over by 3D printers. He plans to solve this by printing an extension, once he gets enough printers.

Above
The LED filaments are amazingly flexible

LEDs are one of the first components people use when they start electronics, and now most houses are lit by LED bulbs. Does that mean it should be easy to make your own lights? Not just building an assembly for light bulbs, but completely from scratch? That was the starting point of this project.

We wanted a filament effect, and it's been possible to purchase LED filament – the sort they use in light bulbs – for a long time now. However, it's difficult to work with. The filament is fragile, and the power supplies are a bit tricky to get right if you've not built a power supply before. More recently, though, a new type of LED filament has come on the market – flexible 3V filament.

Before looking at the two different types, let's first look at what LED filament is. Basically, it's a load of LEDs laid out very closely together in a line so that when you apply power, it looks like the whole strip is glowing uniformly. If you turn the power down so that they're only just lit, you can usually see the individual points of light, but as it gets brighter, these diminish, and shouldn't be visible at full power.

LEDs can be hooked up in different ways. Basically, the power requirements of a strip come down to whether they're hooked up in parallel or in series. Or, more commonly, there will be multiple blocks in parallel where each block is several strips hooked up in series.

As LEDs are semiconductors, they don't obey Ohm's law, and there are a couple of things you need to know about working with them. Firstly, there is a minimum amount of voltage they need to turn on. Secondly, if you apply much more than this, they very rapidly start to conduct too much current and can burn out. Typically, you prevent LEDs getting too much current with a protection resistor, but with high-output LEDs such as those in lamps, this can result in wasted energy. There are a few solutions to this, but the one that we've gone with is just to match the voltage to the LED strip closely. Now, let's go back to looking at the different types of LED filament.

It's got a few advantages of the flexible filament over the old type of ridged filament:

- It's easy to bend into almost any shape you want
- 3V supplies are plentiful and cheap

- It comes in 30 cm strands that give a pleasing effect without having to solder lots of strips together

Of course, there are some disadvantages:

- It's not that bright, so if you want a lot of light, you'll need a lot of strips
- Being low voltage, it's a higher current for the same amount of light

The very simplified difference is that traditional LED filament is good for making light bulbs for lighting rooms. 3 V flexible LED filament is good for making dimmer, softer lighting that's more akin to lamps.

POWER AND ASSEMBLY

Each strip pulls about 120 mA when run at 3 V, and we needed a power supply. It would be easy enough to find one that can power this, but as this is an upcycling project, we decided to rummage through a box of old power supplies to see what we could find. We came across a 12 V 1 A supply that had formerly been some BT doohickey. 12 V is obviously a lot more than 3 V – in fact it's four times more, so we wired four lengths of this LED filament in series. The electrics really were as simple as that.

Keeping with our upcycling theme, we scoured the kitchen and the wood-pile for additional bits and pieces. Four glass ramekins (that once held Gü puddings) served as the protection for the LED filament (sort of akin to the glass bulb on a normal light bulb), a bit of ply held the filaments and glass in place, and a larger chunk of wood acted as the base.

Assembly was just a case of drilling holes for the wires, hot-gluing the ramekins in place (OK, this should really have been done better – we're going to excuse ourselves by calling this a prototype and improve it later, if we find it useful), and screwing the wood to the base.

There you have it. In total, the new parts we bought for this project were just four lengths of LED filament (about £1 each – search your favourite direct-from-China website, and you should find similar options). We also used a bit of hookup wire and solder to join it all together. Everything else was upcycled.

The flexible filament is easy to use, looks awesome, and is perfect for building your own lighting systems. We'd love to see what you create.

THE IMPORTANCE OF UPCYCLING

In the coming issues, we're going to be looking at upcycling a bit more. Waste is a huge issue for our society. So is the availability of electronic parts. While it is convenient to be able to order the precise thing you need and get it delivered to you within a few days, is it really the best option? Reusing old parts also exercises your creativity and brings a unique look to your projects. We could have made this project with new glass or acrylic domes to protect the LED filament, but it wouldn't have had the same aesthetic. Similarly, the saw-marks on the wood just add to the look. Reusing a power supply probably cut the cost of this project in half – or maybe even more.

Left
The finished project provides a soft, warm light. Great for what interior designers call ambience

Below
It's just four LED filaments wired in series – what could be simpler? You could also add a switch, but we turn it off at the wall

TUTORIAL

LED matrix GIF player with WLED

Send animations over the air waves

Ben Everard

@ben_everard

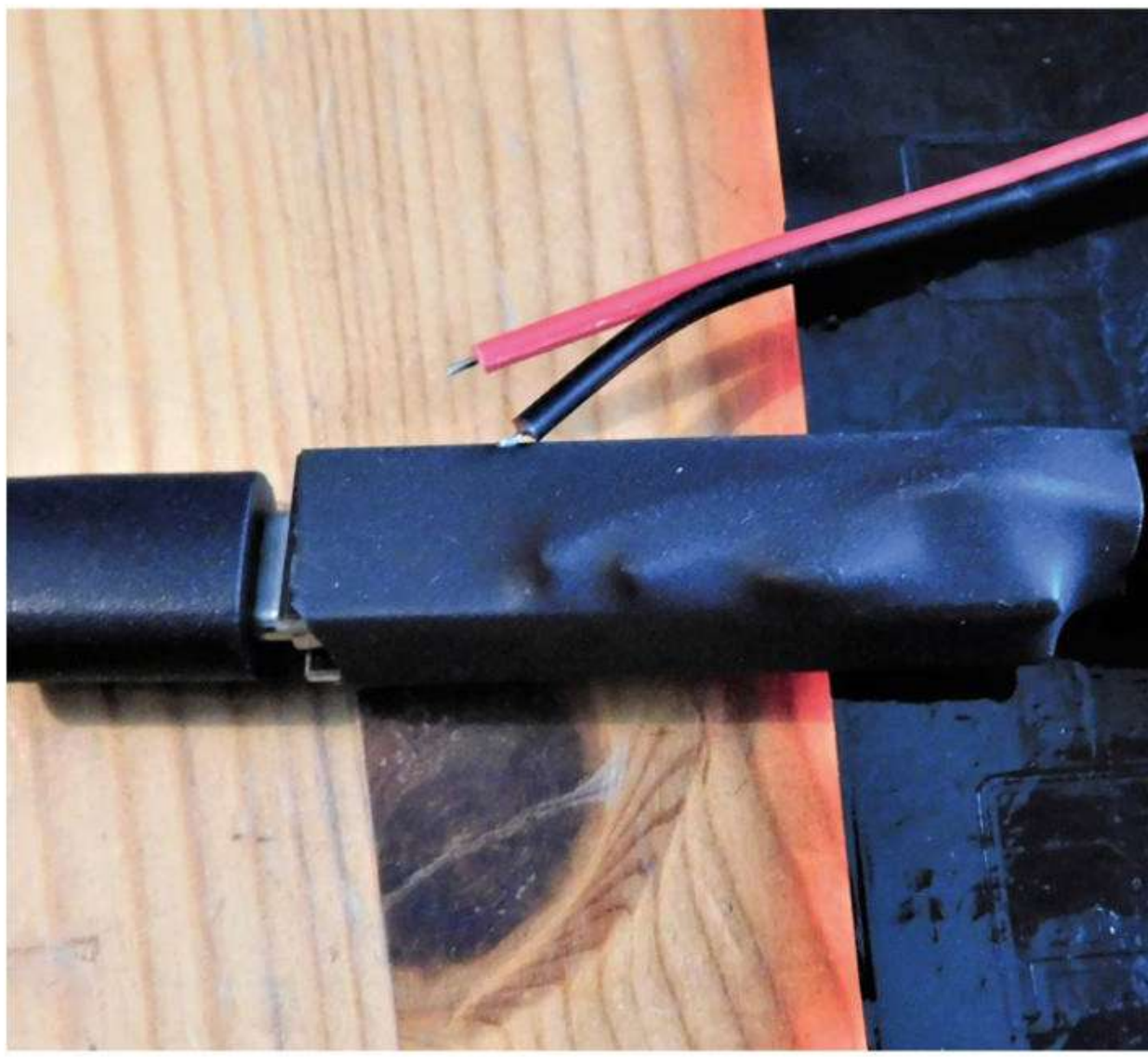
This month, Ben has mostly been pondering the question 'how many LEDs is too many?'

WLED is firmware for ESP32-based devices that lets you easily control RGB LEDs (such as NeoPixels) over WiFi. This can be really useful because it means you don't have to have physical access to the controller to change what is playing. In this article, we're going to connect a 16×16 matrix up to an ESP32 board (the Smôl from SparkFun), load on WLED, and then control it from another computer on the network by playing animated GIFs.

WLED has loads of features and can be used to create all sorts of lighting effects that look wonderful, but that we won't use in this tutorial today. If you're setting up some LED lighting, though, it's well worth browsing the project website, **kno.wled.ge**. One of the really great features is how easy it is to get up and running. You don't need any programming experience or even any programming environment: just point your web browser to: **install.wled.me**. If you plug your ESP32 device into your computer's USB port and click Install, you should end up with WLED installed on your device.

Next, we need to connect our LED matrix. When it comes to LED matrices, there are only two common ones: HUB75 matrices that typically come on stiff PCBs (and are sometimes called Px where x is the

Right
Only simple GIFs work on a 16×16 matrix



spacing of the LEDs in millimetres, such as P5), and WS2812B matrices that usually come on flexible PCBs. WLED only works with the latter of these, so make sure you get the right one.

Connecting your controller to the matrix is pretty simple. You just need to wire 5V to the power input, ground to ground, and a microcontroller pin to the data-in pin. For simple testing, supplying power directly from the microcontroller like this is usually OK (and WLED has power control, as we shall see), but if you want to control a lot of LEDs and have them bright, you'll need to wire up a more capable power source.

With the LEDs connected and the software installed, it's time to connect. When you first start WLED, it will start its own WiFi access point called WLED_AP. This has the password wled1234. If you connect to this, it will take you to a 'captive portal'. That may show up as a 'this network needs you to sign in' message, or it may show up when you try to navigate to a website. Either way, it'll take you to a web page that lets you configure the WiFi settings. If you enter your WiFi SSID and password in there and save, the WLED_AP will shut down and the ESP32 device will try and connect to your local network.

You are now ready to connect to the device, but you need to know the IP address of it. If you have access to your WiFi router's settings, you can take a look at the connected device settings and you should see WLED listed there with an IP address. If you don't, you can install the WLED app on Android or iPhone. This will scan for any connected WLED instances.

Once you've found the IP address, you can connect to your LEDs using either the app or just by typing the IP address into a web browser. There, you can set the colours and add effects.

REMOTE ACCESS

There are a few ways of controlling WLED remotely. There's an API, but this is mostly used for changing between different colours and effects. What we want is a way to remotely control all the LEDs – a bit like using it as a remote screen. For this, we can turn to DMX512. This is a protocol that was designed to control lighting effects. It was originally a wired protocol, but through the E1.31 protocol, it can run over WiFi, and this is what we'll use.

One slight quirk of DMX is that it splits the lights up into 'universes' with each universe holding 512 different values. A value could be anything – it could be the value of a dimmer, it could be used to trigger a sound effect, but for us, it'll be the value of either red, blue, or green on an LED. Since each LED needs three values, that means that we can fit at most 170 LEDs into one universe. Since we're using a 16×16 matrix, we'll need to use two universes. That's not a huge problem, but it's something that we need to remember when sending data.

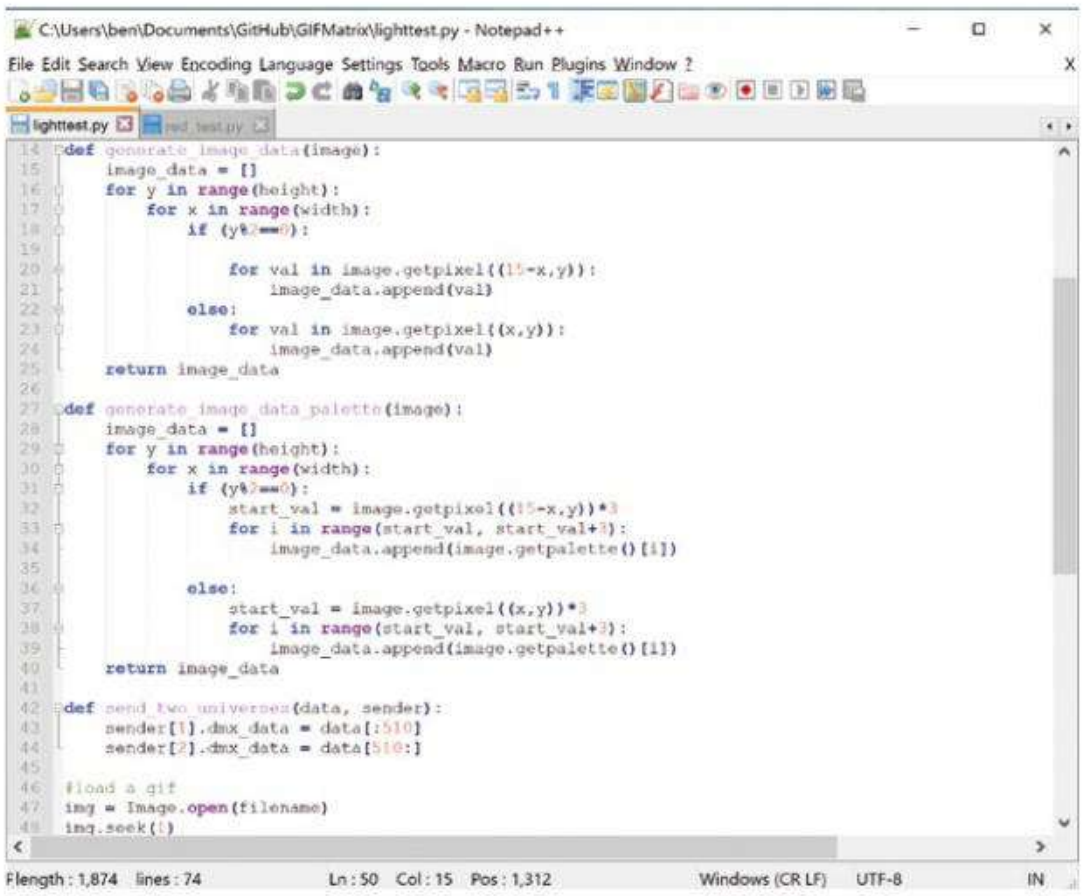
Let's now turn our attention to the software running on the control computer. We've tested this out on Windows, but as far as we can see, there's nothing platform-dependent here, so it should work on Linux (including Raspberry Pi) or macOS as well. We're using Python 3 with two libraries: sacn and PIL (via Pillow). You can install these with:

```
pip3 install sacn
pip3 install pillow
```

Now, let's look at a really simple test – just setting all the pixels red:

```
import sacn
import time

ip_address = "192.168.1.81"

num_leds = 256

#connect to matrix
sender = sacn.sACNsender(fps=40)
sender.start()  # start the sending thread
sender.activate_output(1)
```

Above
Almost any ESP32-based board should work. We used the SparkFun Smôl that we had in to test for the last issue



Above
Animated GIFs on WS2812B LEDs in just 74 lines of Python

```
sender[1].destination = ip_address
sender.activate_output(2)
sender[2].destination = ip_address

data = []
for i in range(num_leds):
        data.append(10)
        data.append(0)
        data.append(0)

def send_two_universes(data, sender):
   sender[1].dmx_data = data[:510]
   sender[2].dmx_data = data[510:]

send_two_universes(data, sender)

time.sleep(5)
sender.stop()
```

There are a few decisions we've made in this code. The first is to connect to a specific device. This isn't actually necessary. E1.31 senders often send data multicast to every device on a network and then use the universe numbers for making sure only the right things light up, but Windows doesn't really like doing that, so we're using a specific IP address instead. As you can see, we've set a destination for each universe and activated it. We've then created a set of data that's just the RGB colour 10,0,0 over and over again. We could do something a bit more artistic, but this is just a test and the artistic bit comes next. We've used Python's list slicing to extract the data for the first and second universes, and then sent them.

WLED will only display this while the E1.31 connection remains open. In this case, we've paused for five seconds and then stopped the connection, which will revert your LED matrix back to whatever it was doing before.

If you run this, hopefully your matrix will light up red for five seconds.

You could use this to implement all sorts of effects. Let's now add in Pillow to display an animated GIF.

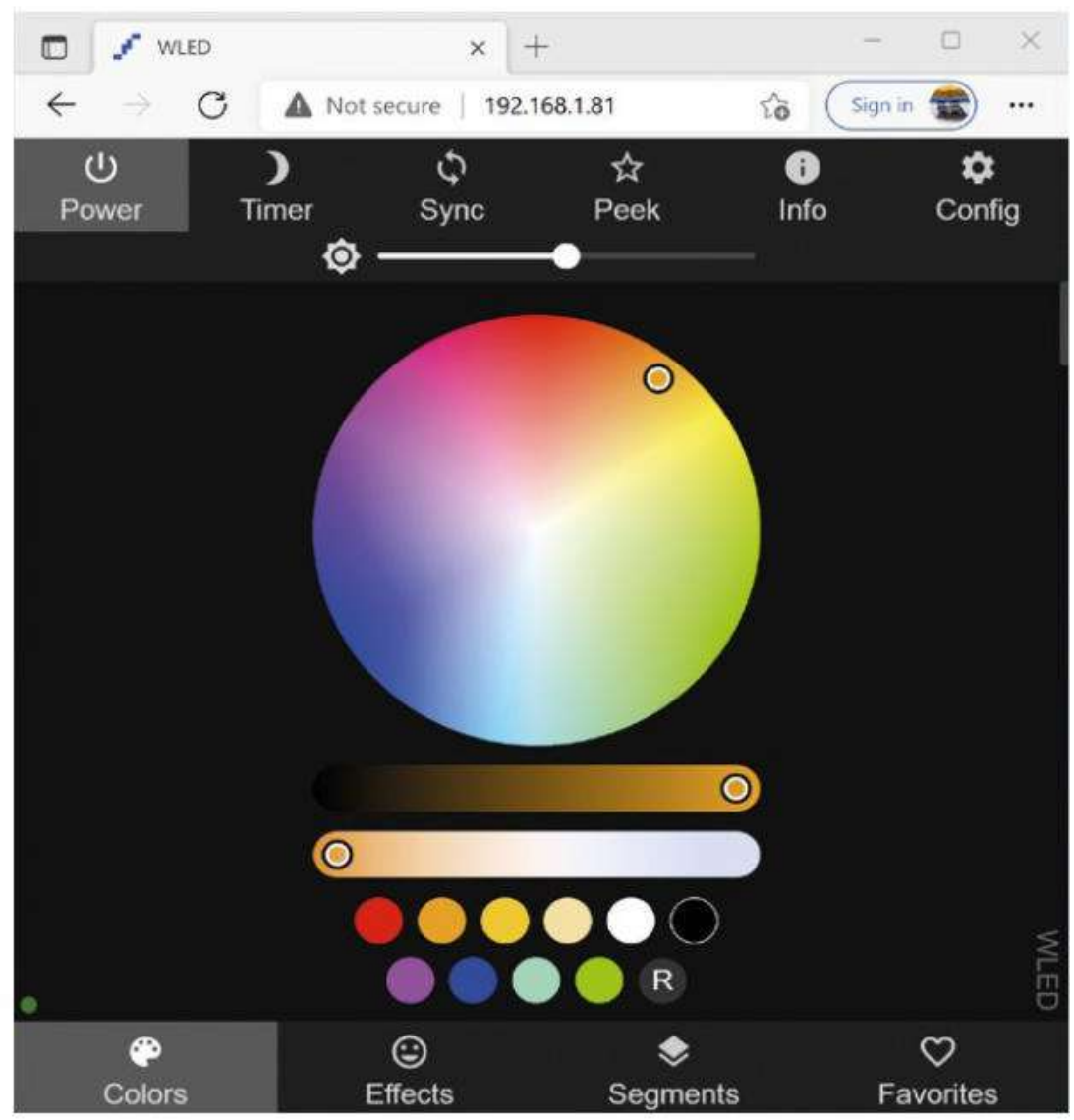
```
import sacn
import time
time.sleep(5)
from PIL import Image

ip_address = "192.168.1.81"
filename = "fire.gif"

width=16
height = 16
matrix_size = (width,height)

frame_pause = 0.5

def generate_image_data(image):
    image_data = []
    for y in range(height):
        for x in range(width):
            if (y%2==0):

                for val in image.getpixel((15
x,y)):
                    image_data.append(val)
            else:
               for val in image.getpixel((x,y)):
                    image_data.append(val)
    return image_data

def generate_image_data_palette(image):
   image_data = []
    for y in range(height):
        for x in range(width):
            if (y%2==0):
                start_val = image.getpixel((15-
x,y))*3
                for i in range(start_val, start_
val+3):
                    image_data.append(image.
getpalette()[i])

            else:
                start_val = image.
getpixel((x,y))*3
                for i in range(start_val, start_
val+3):
```

```
                    image_data.append(image.
getpalette()[i])
    return image_data

def send_two_universes(data, sender):
    sender[1].dmx_data = data[:510]
    sender[2].dmx_data = data[510:]

#load a gif
img = Image.open(filename)
img.seek(1)
out_imgs = []
out_datas = []
try:
   while 1:
      this_img = img.resize(matrix_size)
      out_datas.append(generate_image_data
palette(this_img))
      img.seek(img.tell() + 1)
except EOFError:
   pass

#connect to matrix
sender = sacn.sACNsender(fps=40)
sender.start()  # start the sending thread
sender.activate_output(1)
sender[1].destination = ip_address
sender.activate_output(2)
sender[2].destination = ip_address
sender.manual_flush = True

for i in range(5):
    for data in out_datas:
        send_two_universes(data, sender)
        sender.flush()
        time.sleep(frame_pause)

sender.stop()
```

First, we have to load in the data from the animated GIF. Pillow is really designed to work with static images, but you can use image sequences. Basically, the image object will act like a static image, but you can use the **seek** method to change the particular static image in the sequence that it will use. In this code, we resize any image to 16×16, regardless of whether that makes any sense. It then extracts the colours. There are two methods here for extracting colours. On some images (such as JPEGs), the **getpixel()** method will return R, G, B values for that particular pixel. In GIFs, it instead returns a colour number that references the image palette, which is an index of the colours used in the image. We've included both methods here to help, but only use the palette version with GIFs. We also have to re-sequence the data because our matrix is wired in a zig-zag pattern, with each row going in an opposite direction. Typically, when you set a universe's **dmx_data**, it will send that instantly, but if you have more than one universe, this could cause tearing of the image. Instead, we're setting manual flush, which means that both parts of the image will be set at the same time. That's our basic method of displaying an animated GIF on an LED matrix. You should be able to use the same basic method for displaying all sorts of images and effects on WS2812B LED matrices.

TIMINGS

DMX512 was designed for stage lighting, and for this you need the timings to be very precise. It's used to synchronise sound and lighting effects with awesome precision. However, this is based on using wired connections. Once you go wireless, it's very hard to have accurate timing control. For many purposes, the convenience of WiFi will override any slight issues with timings, but if timings are critical, you'll want to consider using a wired setup.



Below
The WLED firmware has a web page allowing you to customise different parts of the display

Part 03

Explore the sensory world: Build a simple weather station

MAKER

Phil King

Long-time contributor to *The MagPi*, Phil is a freelance writer and editor with a focus on technology.

@philkingeditor

Measure temperature, humidity, and UV light with this basic weather station

Previously in this series, we have built a couple of alarms: one for fire and gas safety, the other for detecting intruders. This time we're doing something a little different by using a couple of sensors to measure certain weather conditions: temperate, relative humidity, and ultraviolet light. One of the sensors gives an analogue output, so we'll learn how to convert that to a digital reading using an ADC.

You'll Need

- DHT11 sensor **magpi.cc/dht11**
- UV sensor **magpi.cc/uvsensor**
- MCP3008 ADC **magpi.cc/mcp3008**
- Jumper wires

01 Connect DHT11 sensor

The DHT11 sensor measures temperature and relative humidity. We're using one from the Waveshare Sensors Pack, available from The Pi Hut (**magpi.cc/wavesensors**), and also sold separately. You can buy a similar DHT11 sensor elsewhere, sometimes without the PCB, or you could upgrade to a DHT22 or use a different sensor such as a BME280 (which also reads barometric pressure).

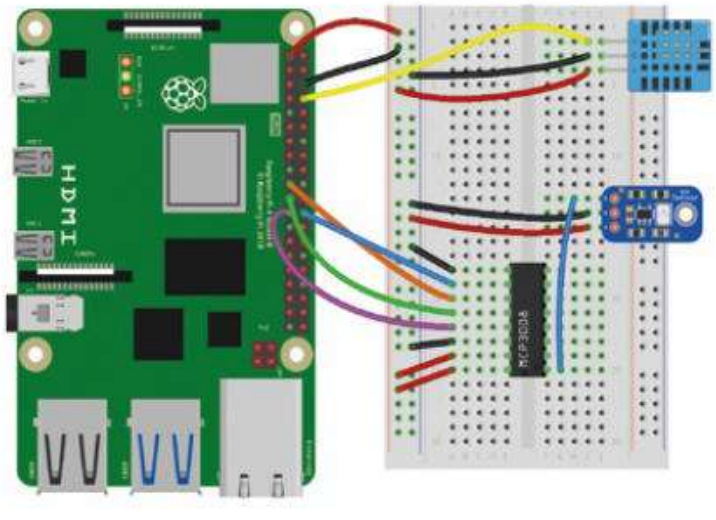


Figure 1 The wiring diagram for the weather station, including the DHT11 sensor, UV sensor, and ADC

The DHT11 combines a digital thermistor with a capacitive humidity sensor and outputs digital readings, so it's easy to use. With the power turned off, connect the DHT11 sensor to Raspberry Pi as in **Figure 1**. We're powering it from Raspberry Pi's 3V3 pin, grounding it with a GND pin (both via the breadboard side rails), and the digital output (marked DOUT on the sensor) is going to GPIO14.

02 Install DHT11 library

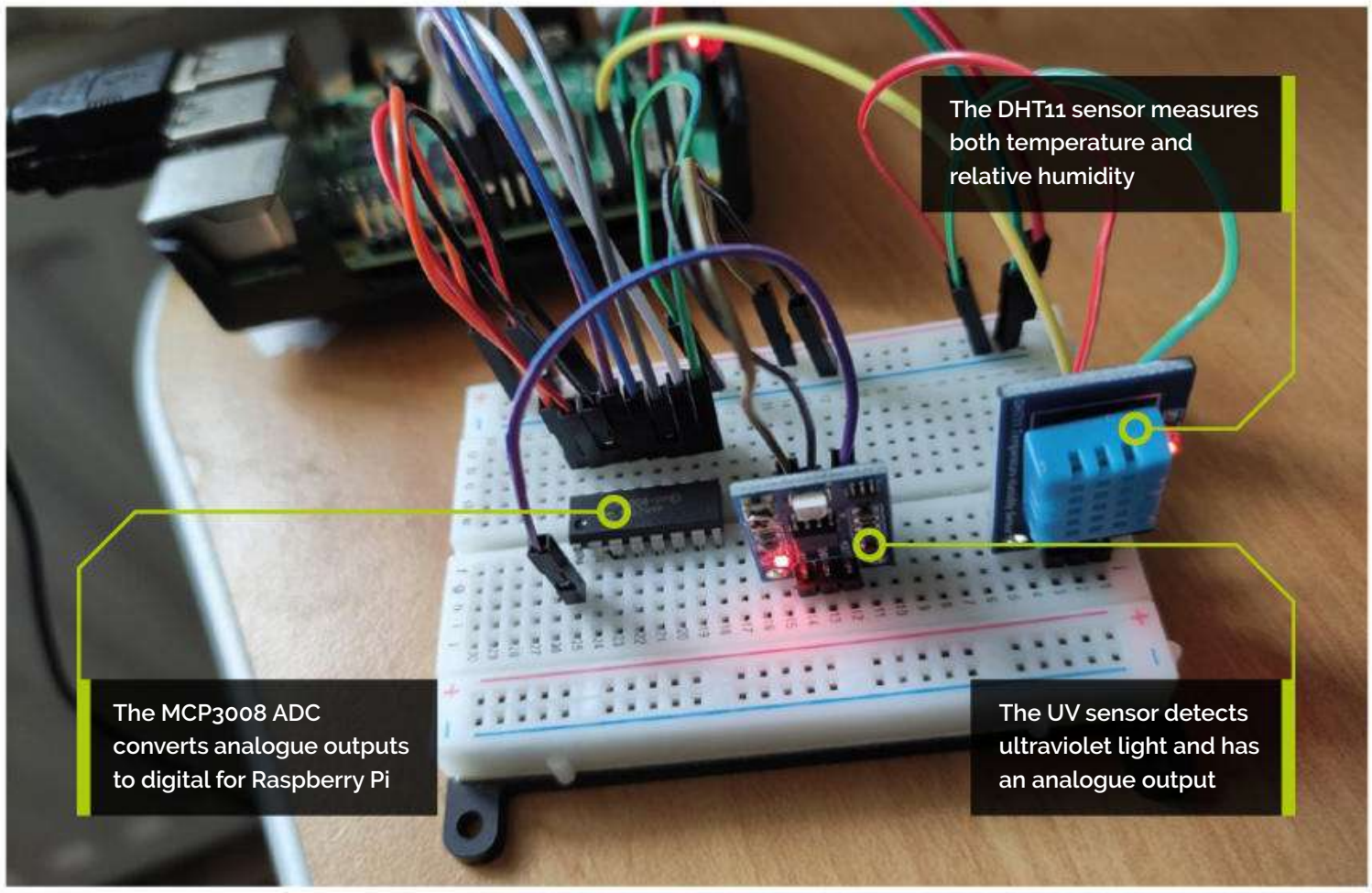
With the DHT11 sensor wired up, turn on Raspberry Pi. You should see the sensor's red power LED (on the right) light up.

Since the DHT11 outputs its data in binary form to the GPIO14 pin, we'll need a way of converting that into decimal numbers. The easiest way is to use a ready-made Python library. We're using szazo's DHT11 library, which can be installed with the following Terminal command:

```
pip3 install dht11
```

03 Temperature and humidity test

To begin, we'll create a simple Python program (**dht11_test.py**), to read the sensor's temperature and humidity outputs. From the desktop menu, go to Programming > Thonny IDE.



As our DHT11 library uses the RPi.GPIO Python library, we import that as 'GPIO' at the top of the code. We then initialise the GPIO and do a standard clean-up to reset all pins to inputs.

Within an infinite `while True:` loop, we read an instance of the GPIO 14 pin value. If the result is valid (i.e. not an error), we print the temperature and humidity values to the Shell. In our print statement, the `%-3.1f` format parameter sets each output to a minimum three digits including one decimal place. We add the end `= "\r\"` parameter so the message is always printed on the same line.

Run the **dht11_test.py** code and check that you are getting realistic readings. Try breathing on the sensor to increase both temperature and humidity.

04 Add an ADC

The Waveshare UV sensor (GUVA-S12SD) we're using outputs an analogue signal, so we'll need to convert that to a digital value using an ADC (analogue-to-digital converter). We're using the MCP3008 ADC, which has eight input channels.

Since this chip uses the SPI interface, we'll need to enable SPI in the Raspberry Pi Configuration tool. It's also best to enable full SPI support in Python 3. To do so, open a Terminal window and enter:

```
sudo apt-get install python3-spidev
```

With the power to Raspberry Pi turned off, it's time to wire up the ADC. Place the MCP3008 in the middle of the breadboard, straddling its central groove. Make sure it's the correct way round, as shown in **Figure 1**, with the top of writing on top of the ADC nearest Raspberry Pi.

Now connect the jumper wires as in **Figure 1**. Two of them go to the '+' breadboard power rail, connected to a 3V3 pin; two others are connected to a GND pin via the '–' rail. The four middle legs of the ADC are connected to the SPI interface on Raspberry Pi: GPIO pins 8 (CE0), 10 (MOSI), 9 (MISO), and 11 (SCLK).

dht11_test.py

Language: **Python 3**

```
001.  import RPi.GPIO as GPIO
002.  import dht11
003.
004.  # intialise GPIO
005.  GPIO.setwarnings(False)
006.  GPIO.setmode(GPIO.BCM)
007.  GPIO.cleanup()
008.
009.  while True:
010.      instance = dht11.DHT11(pin = 14)
011.      result = instance.read()
012.      if result.is_valid():
013.          print("Temperature: %-3.1f C" % result.temperature,
      "Humidity: %-3.1f %%" % result.humidity, end = "\r")
```

uv_test.py

Language: **Python 3**

```
001.  from gpiozero import MCP3008
002.
003.  uv = MCP3008(0)
004.
005.  while True:
006.      print("UV: %-3.5f V" % (3.3 * uv.value), end = "\r")
```

uv_index.py

Language: **Python 3**

```
001.  from gpiozero import MCP3008
002.
003.  uv = MCP3008(0)
004.
005.  def uv_range():
006.      global uv_mv
007.      global uv_index
008.      uv_mv = int(3300 * uv.value)
009.      if uv_mv in range (0,227):
010.          uv_index = 0
011.      elif uv_mv in range(227,318):
012.          uv_index = 1
013.      elif uv_mv in range(318,408):
014.          uv_index = 2
015.      elif uv_mv in range(408,503):
016.          uv_index = 3
017.      elif uv_mv in range(503,606):
018.          uv_index = 4
019.      elif uv_mv in range(606,696):
020.          uv_index = 5
021.      elif uv_mv in range(696,795):
022.          uv_index = 6
023.      elif uv_mv in range(795,881):
024.          uv_index = 7
025.      elif uv_mv in range(881,976):
026.          uv_index = 8
027.      elif uv_mv in range(976,1079):
028.          uv_index = 9
029.      elif uv_mv in range(1079,1170):
030.          uv_index = 10
031.      elif uv_mv >= 1170:
032.          uv_index = 11
033.
034.  while True:
035.      uv_range()
036.      print("UV index: ",uv_index, end = "\r")
```

05 Connect the UV sensor

With the ADC wired up to Raspberry Pi, we can now add our UV sensor to the setup. As in **Figure 1**, we connect its VCC pin to 3.3V via the breadboard power rail, and its GND pin to GND on Raspberry Pi via the breadboard ground rail.

Finally, we connect the sensor's AOUT (analogue out) pin to the MCP3008's channel 0 pin, as shown in **Figure 1**. You could wire it to any of the eight channels on that side, but we're using this one in our code.

06 UV test

Let's create a simple program to test the sensor. In the **uv_test.py** listing, this time we're using the GPIO Zero library as it has a handy MCP3008 class, which we import at the top. We assign the `uv` variable to the MCP3008's channel 0 to read the connected sensor's analogue output.

In a `while True:` loop, we multiply the digitally converted sensor output (which ranges from 0 to 1) by the 3.3 maximum voltage to get an accurate voltage reading. If you're indoors, it should be very low. To test it, you could move the sensor outside, or just hold it through an open window. Alternatively, you could try shining an ultraviolet light (such as from a counterfeit currency detector) onto it. You should see the output value change.

07 UV index level

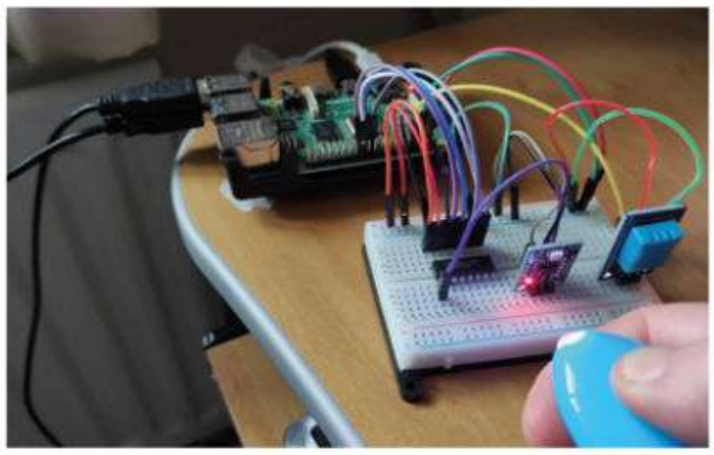
We're getting an accurate voltage reading from our UV sensor, but what does it mean? We need to convert it to the international standard UV index for it to be useful.

In the **uv_index.py** listing, we create a `uv_range` function with a series of `if` and `elif` statements to check which range of values the reading (converted into microvolts in the `uv_mv` variable) falls into and then set the UV index level (`uv_index`) accordingly.

Note: We've estimated the ranges for this based on a table of typical voltage values for each index that we found online for the sensor, but it may vary depending on temperature.

08 Bringing it together

Finally, let's combine the code with our earlier DHT11 program to read both sensors and print the appropriate weather information, in the



Testing the UV sensor with ultraviolet light. It needs to be of the correct wavelength to be sensed

weather.py listing. We can use both the RPi.GPIO and GPIO Zero libraries (the latter is based on the former), but we need to initialise the GPIO pins before setting any GPIO Zero values.

We include our `uv_range` function to determine the UV index and add a call to it in the `while True:` loop. We also add its output to the print function.

09 Take it outside

If you want to locate your weather-sensing project outside, you'll need a weatherproof case for the electronics. Alternatively, you could even house it in a couple of connected drainpipe bends, as we used for an air sensor project in *The MagPi* issue 92 (**magpi.cc/92**) – in which case, it's best to use a Raspberry Pi Zero W to save space.

You'll need to power the project via an extended USB cable. Assuming you want to read the outputs from it remotely, via SSH or possibly a web dashboard, you'll also need to connect Raspberry Pi to your router – if doing so wirelessly, depending on the project's distance from the router, you may need a WiFi range extender.

Taking it further

We have made a very basic weather station. To improve it, you could add a barometric pressure sensor such as the BMP280 – or switch out the DHT11 for a BME280, which measures temperature, humidity, and pressure, but you'll need a different Python library for that.

If locating it outdoors, you could also add special weather sensors such as an anemometer (wind speed), rain gauge, and wind vane, as featured in the SparkFun SEN-15901 Weather Meter Kit (**magpi.cc/sfweatherkit**).

Next time we'll use moisture and liquid level sensors to monitor a plant. See you then.

weather.py

DOWNLOAD THE FULL CODE:
magpi.cc/github

> Language: **Python 3**

```
001.  import RPi.GPIO as GPIO
002.  import dht11
003.  from gpiozero import MCP3008
004.
005.  # intialise GPIO
006.  GPIO.setwarnings(False)
007.  GPIO.setmode(GPIO.BCM)
008.  GPIO.cleanup()
009.
010.  uv = MCP3008(0)
011.
012.  def uv_range():
013.      global uv_mv
014.      global uv_index
015.      uv_mv = int(3300 * uv.value)
016.      if uv_mv in range (0,227):
017.          uv_index = 0
018.      elif uv_mv in range(227,318):
019.          uv_index = 1
020.      elif uv_mv in range(318,408):
021.          uv_index = 2
022.      elif uv_mv in range(408,503):
023.          uv_index = 3
024.      elif uv_mv in range(503,606):
025.          uv_index = 4
026.      elif uv_mv in range(606,696):
027.          uv_index = 5
028.      elif uv_mv in range(696,795):
029.          uv_index = 6
030.      elif uv_mv in range(795,881):
031.          uv_index = 7
032.      elif uv_mv in range(881,976):
033.          uv_index = 8
034.      elif uv_mv in range(976,1079):
035.          uv_index = 9
036.      elif uv_mv in range(1079,1170):
037.          uv_index = 10
038.      elif uv_mv >= 1170:
039.          uv_index = 11
040.
041.  while True:
042.      uv_range()
043.      instance = dht11.DHT11(pin = 14)
044.      result = instance.read()
045.      if result.is_valid():
046.          print("Temperature: %-3.1f C" % result.temperature,
      "Humidity: %-3.1f %%" % result.humidity, "UV index: ",
      uv_index, end = "\r")
```

FreeCAD, summing up!

In this final part of the FreeCAD series, we look at where you may like to explore next as part of your ongoing FreeCAD journey!

Jo Hinchliffe

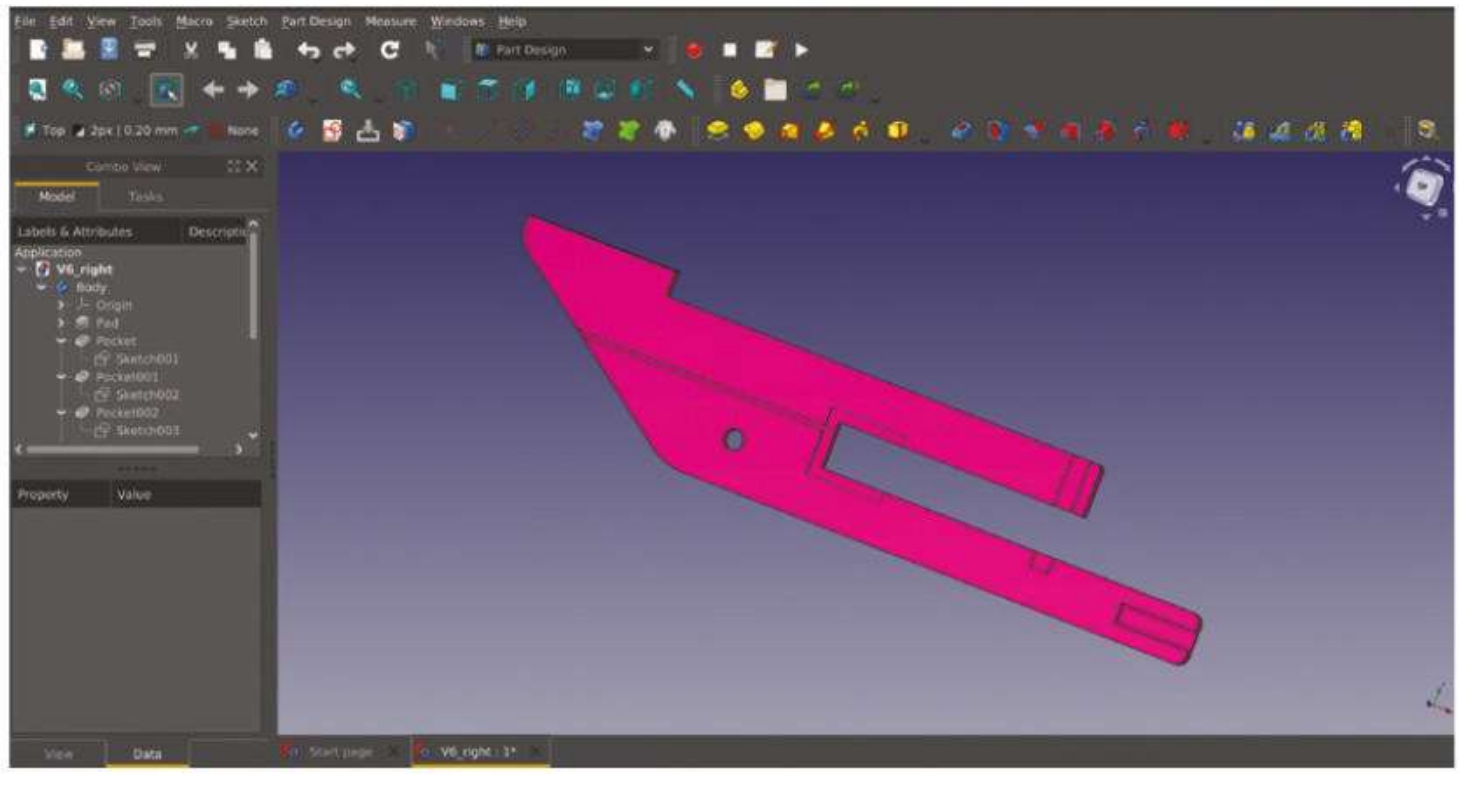
@concreted0g

Jo Hinchliffe is a constant tinkerer and is passionate about all things DIY space. He loves designing and scratch-building both model and high power rockets, and releases the designs and components as open-source. He also has a shed full of lathes and milling machines and CNC kit!

In this sixteenth, and final, part of the FreeCAD series, we thought it might be a good idea to look at some areas that you might like to explore now that you have a good grasp of the basics. You probably have some intermediate skills if you have followed through this series fully, so let's look at other areas of FreeCAD that you might explore.

For the sake of consistency throughout this series, we have used FreeCAD in its most default appearance – this makes it easier for people to recognise where tools are from the images etc. However, it's possible to change up the theme and even change the user interface entirely. For simple changes of theme go to Edit > Preferences and then click the General tab. You should see options to change the 'stylesheet', as well as options to change the size of the toolbar icons. In **Figure 1**, we have changed the stylesheet to 'Darker Orange', and increased the tool size to medium.

For a more significant change, you might like to try a different user interface. This can be achieved by installing the 'ModernUI' workbench from the Tools > Addon menu. You'll be prompted to restart FreeCAD and will be rewarded with a different UI where one of the major differences is that the workbenches are tabbed across the top of the screen, allowing for quick switching (**Figure 2**). If you don't like the ModernUI, you can uninstall it in the usual way using the Addon manager but, additionally, you need to create and execute a macro to return to the original FreeCAD UI. This is straightforward to achieve – you select and copy the macro script text that is supplied in the description of the ModernUI workbench in the Addon manager. Then, from any workbench in FreeCAD, click the Macro drop-down.



YOU'LL NEED

A computer with FreeCAD 0.19

Figure 1 Changing the stylesheet options in FreeCAD can customise the colours and also change the tool icon sizes

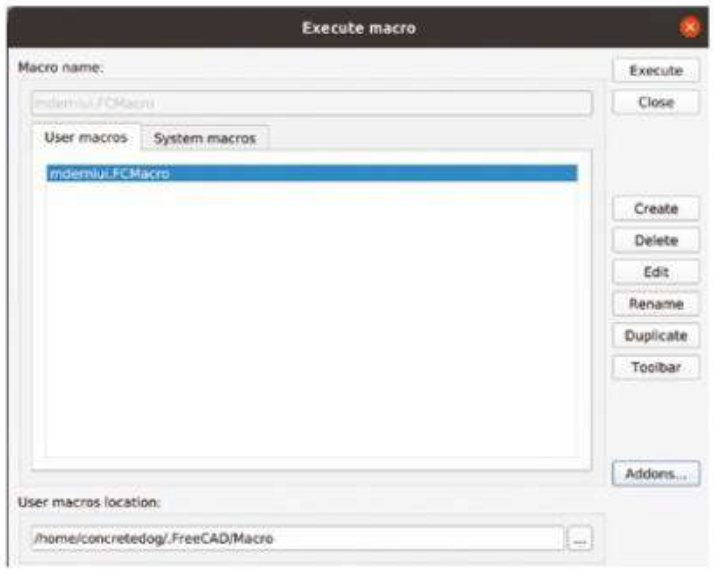


Figure 3
The Macro tools can be used to create simple and complex automated processes alike

In turn, click Macros and, in the Execute macro window, click Create (**Figure 3)**. Paste in the macro script copied from the Addon manager, and then close the tab and click Save and give it a name when prompted. Return to the Execute macro window, highlight the macro you created, and click Execute to run the macro. Then, restarting FreeCAD, you'll have returned your installation to the original state.

Whilst on the subject, macros in FreeCAD can be incredibly useful. If you have regular tasks that you do repeatedly, or similar objects that you create across a range of projects, a macro may help to streamline your workflow. You don't have to script a macro by hand either, you can create macros using the record function. Essentially, once a macro is recording, it builds a script of all the actions you perform until you stop the macro recording. You can then save and name the macro, and execute the macro whenever needed via the macro menu. A simple way to try this is to create a new sketch and then start the macro recorder. Draw a couple of circles and constrain them, and then perhaps Pad them using part design, and then stop the macro. You can look over the script and execute it to repeat the same task. Often macros require a little editing and tweaking. It's been described to us more than once that macros are the gateway to writing your own workbench!

> **Essentially, once a macro is recording,** it builds a script of all the actions you perform until you stop the macro recording

We've come to the opinion, whilst exploring FreeCAD, that it really is unique as a CAD environment and we love that it isn't trying to be an open-source clone of another tool. This becomes really clear when you look at the range of workbench environments there are for FreeCAD. There are so many we haven't explored as →

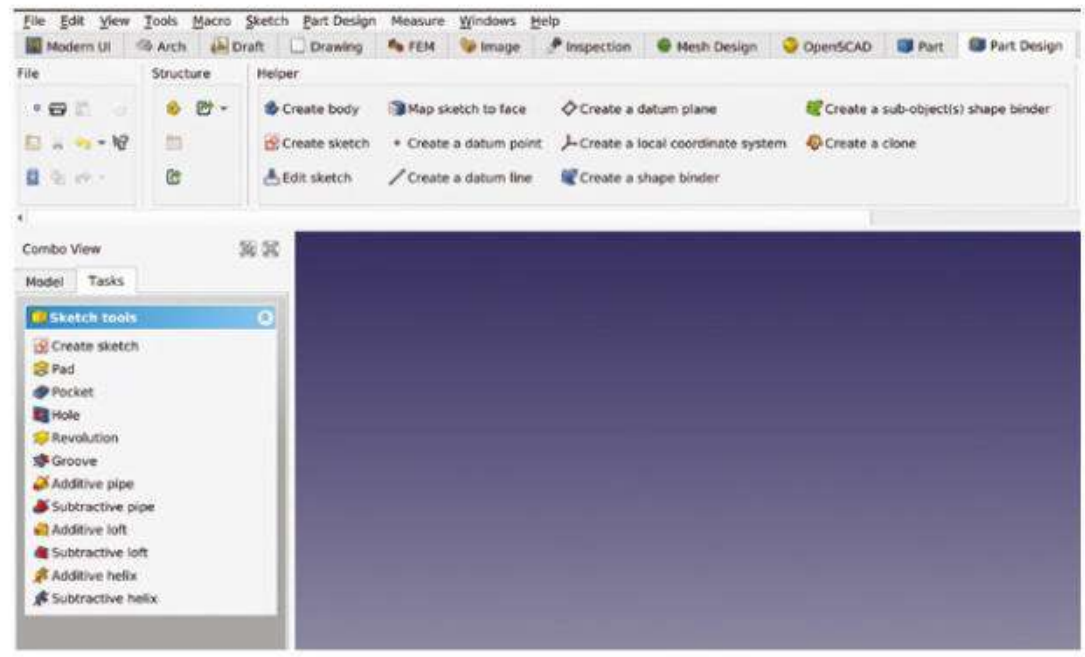
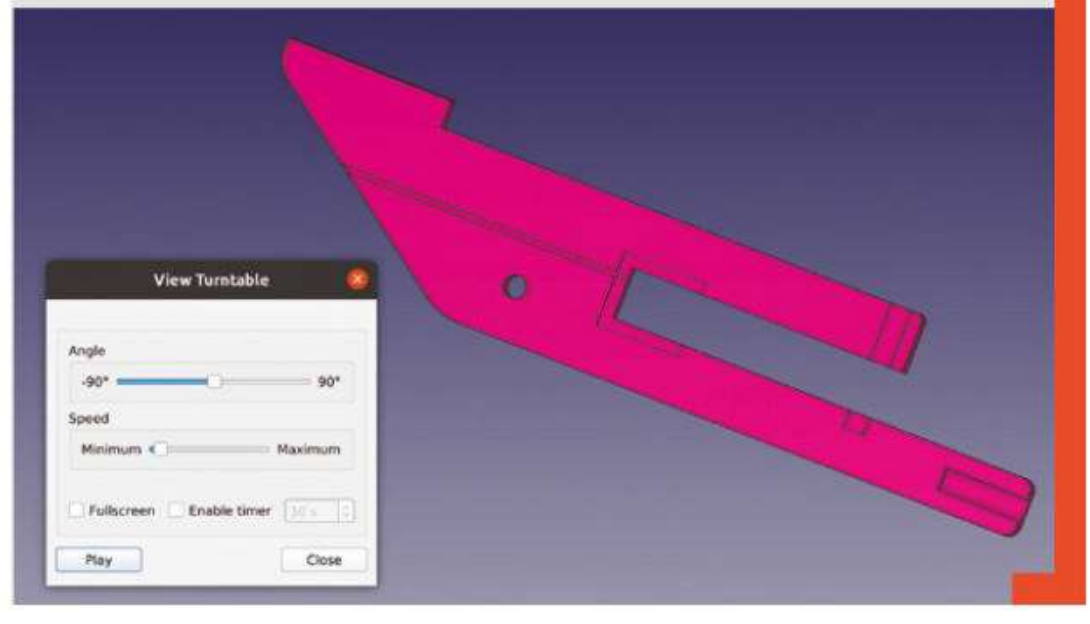
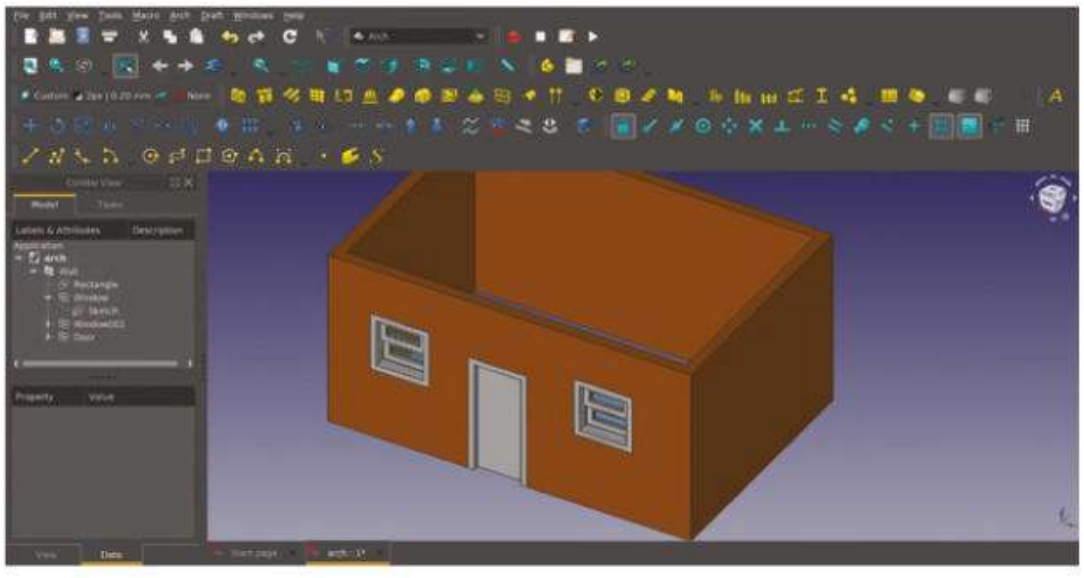


Figure 2
Available via the Addon manager, Modern UI changes the design layout of FreeCAD and uses tabs for each installed workbench

ROTATING

Sometimes you might want to make your 3D model move, and, as ever in FreeCAD, there are numerous ways to achieve this. At first glance, you might jump to installing the Animation workbench from the Addon manager, but currently this only supports older Python 2-based versions of FreeCAD. It's worth keeping an eye out for if it gets any updates in the future. A current popular approach to animating assemblies is to use the Assembly 4 workbench. We haven't looked at this in the series, but we did look at an assembly workbench, A2plus, in issue 43. Whilst you are exploring and learning these approaches, a super-simple way to create a rotating movement of your part or project is by using the turntable tool. Located at Tools > View Turntable, when you launch the turntable dialog, you'll see a couple of simple parameters: you can adjust the angle of the view of the part and you can set the speed of rotation. You can also set the view to full screen and also set a timer for the length of time the part rotates for. Click the play button and your rotating part springs into life. You can use the turntable function in combination with a screen recorder application to make rotating video clips.





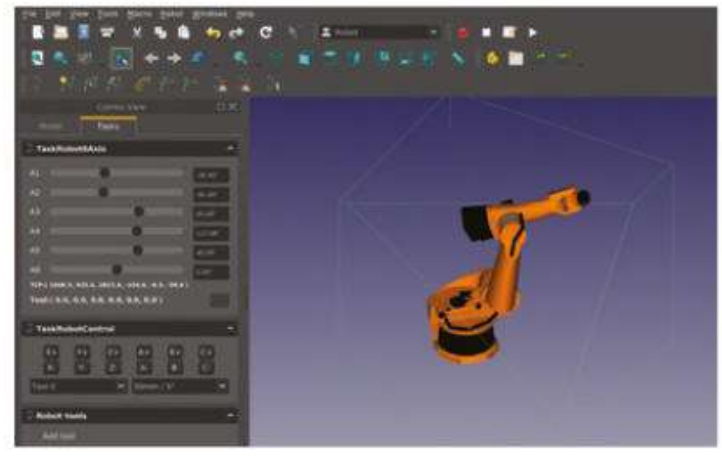


Figure 4
A small, simple example beginning to explore the Architecture workbench tools

Figure 5
An amazing workbench that enables commercial robot arms to be placed and controlled

part of this series, but we wanted to highlight a few examples that you might like to go on to explore.

Perhaps the most obvious one we haven't looked at is the Arch workbench which is built into FreeCAD, usually at the top of the workbench list. Arch is short for Architecture, and it's chock-full of useful tools to help you design buildings and structures. Whilst you may well need to look through the wiki page or find an online tutorial, you can quite quickly get going with this fabulous workbench. In **Figure 4**, we have drawn a simple rectangle using the draft tools in the Arch workbench, and then used a tool to generate a wall from the wire outline. We've then used the built-in tools to simply click and add preset windows and doors in our simple structure. Don't be fooled by our first attempt, though: check out some of the amazing structure designs in Arch on the FreeCAD forums to get properly inspired.

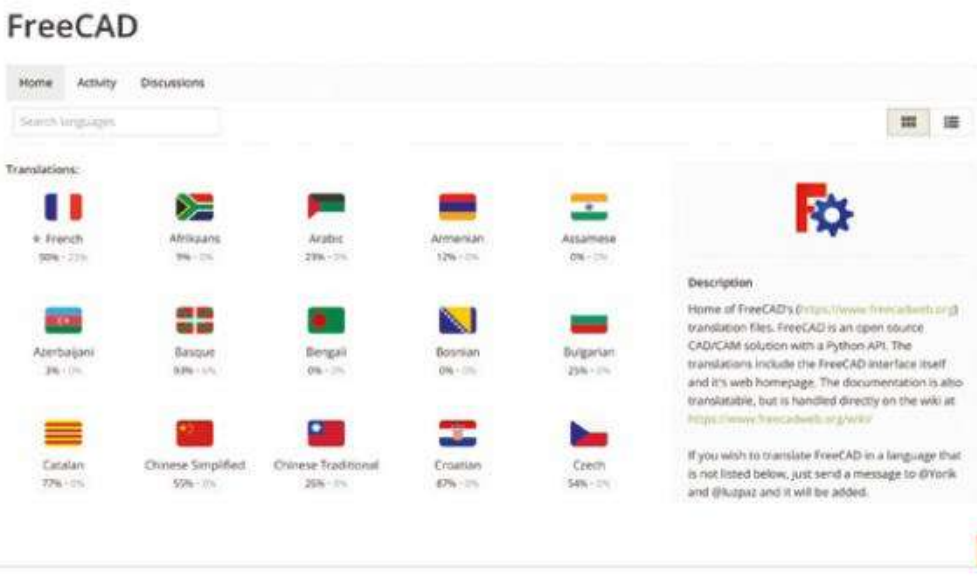
The Robot workbench is perhaps a great example of just how powerful a tool FreeCAD can be. Whilst the Robot workbench is still included in FreeCAD, it isn't being particularly actively developed, but it's a fun workbench to play with for a few minutes. Often, with many workbenches, the FreeCAD wiki has a succinct tutorial to work through to get you up and running with the basics of the bench. The Robot workbench is no exception, and working through the instructions at **hsmag.cc/RobotTutorial**, you can import an industrial robot arm. In **Figure 5**, we used a Kuka IR500, and you can create trajectories to move the arm through, which then can be played like a script. If you happen to own or have access to a Kuka robot arm in real life, you can even export the subroutines you create to run on the machine.

The Path workbench is FreeCAD's main CAM environment, and is where you can set up CNC routing and milling toolpaths to create parts on a wide variety of machines. We wrote about the Path workbench outside of this FreeCAD tutorial series way back in issue 25. Of course, FreeCAD has developed a lot since then and the Path workbench, with its amazing dedicated developers, is one of the most actively developed areas of FreeCAD. As such, it's changed somewhat since our write-up, but there's plenty of information about it online. It's definitely worth exploring, and offers robust and powerful CAM approaches, and is capable of creating G-codes post-processed to suit a wide variety of machines (**Figure 6**).

The FEM workbench enables complex analysis of objects under loads, and this type of FEM/FEA approach often is quite expensive to perform in other proprietary CAD environments. It requires a couple of extra external packages to be installed/configured, and then models can be assigned material properties and then tested under loads. To get a fleeting

CROWD IN

One of the many ways in which people can contribute to the FreeCAD community is by helping translate FreeCAD so it can be used in many different languages. This is a large task but is being well achieved due to the use of 'Crowdin'. The Crowdin website really simplifies crowd-sourced translation. You simply set up an account, click a language you want to help with, and you are presented with every part of FreeCAD text split into little areas. Click an area and then try and translate a few terms or sentences. You can also view and comment on other people's translation attempts which means, with enough people, the translation becomes more and more accurate. This peer reviewing aspect of the Crowdin platform also means that your translation doesn't have to be perfect – often there is no perfect translation and, as such, all attempts help inform the best option. Once a language is translated, the FreeCAD developers can then pull that language into the platform for an upcoming release. Check out the FreeCAD Crowdin page here: **crowdin.com/project/freecad**.



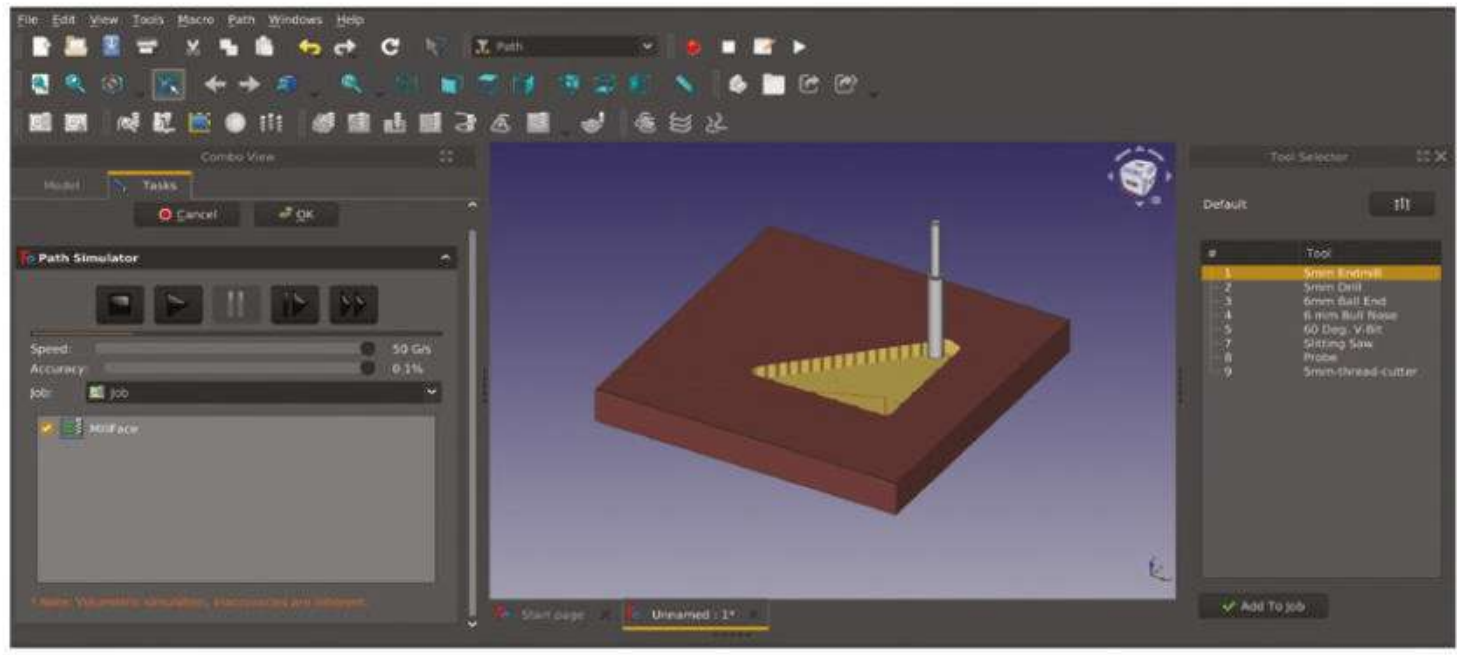


Figure 6
The Path workbench is a powerful CAM tool enabling toolpaths and G-codes to be created and simulated for CNC machines

appreciation of what this looks like, even without installing the extra packages, you can open one of the example projects bundled into FreeCAD on the FEM workbench, and see the type of strain/stress analysis and representation you can achieve (**Figure 7**).

When we looked at the Mesh workbench in issue 48, we needed to have another CAD environment, OpenSCAD, installed on our machine to enable some of the Mesh workbench features. There's also, however, a dedicated OpenSCAD workbench within FreeCAD. OpenSCAD differs from many CAD packages in that it is script-based. This means you 'code' your CAD models. For example, to make a cube in OpenSCAD, you might write a line that looks like '`cube([10,10,10],true);`' which creates a 10 mm cube centred around the origin point. The addition of a dedicated OpenSCAD workbench means that if (as we did) you have used OpenSCAD and are moving to FreeCAD, any components or models you have made in OpenSCAD can still be used within your FreeCAD designs. The workbench also allows you to create new parts using OpenSCAD scripting directly (**Figure 8**). If you've solved a really hard geometry in OpenSCAD, this gives you not only the option of reusing that work, but also for that work to be enhanced using all of FreeCAD's other tools.

Whilst there are more than enough workbenches to get to grips with listed in Addon manager, there are also workbenches out in the community that aren't included either in the main download or in the Addon manager. These are often in development and experimental in nature, but it can be fun if you find something in the community that relates to something you know about or are interested in.

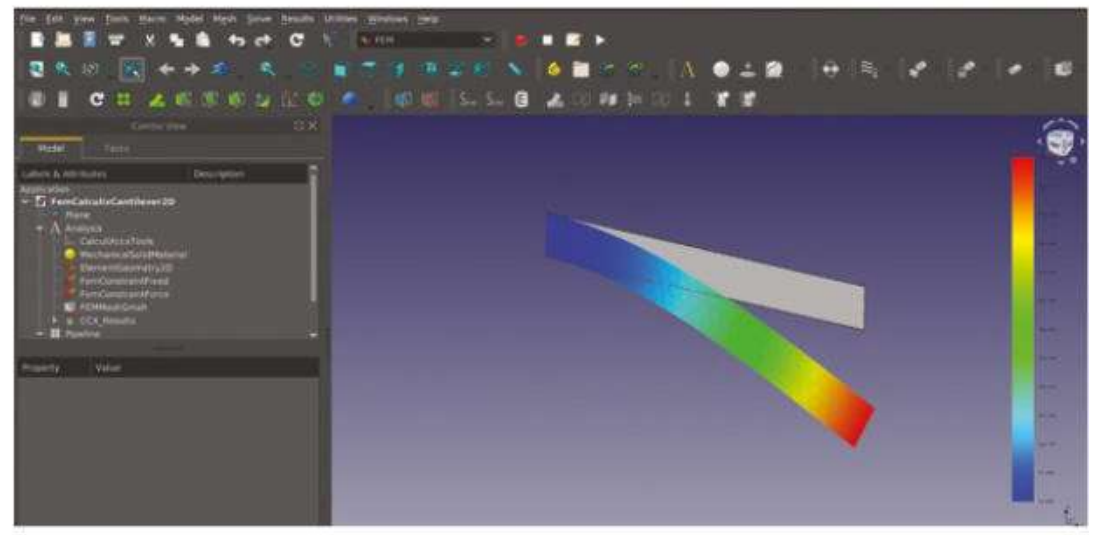


Figure 7
One of the included FEM workbench examples. This example analyses the deflection of a beam under load

> Perhaps you have your own idea for a tool that you'd like to create for FreeCAD, **or an issue that you think you can solve**

Sometimes these workbenches are not included as development on them has ceased. However, often the open licences of these archived workbenches mean that someone could begin to work on them again. They also may still be functional and of use. One excellent example is a workbench first released back in 2015: the Glider workbench enables the design of paragliding canopies and recently, in 2021, a forum member posted pictures of a paraglider in flight that had been designed with this innovative workbench (**Figure 9**).

Are you a developer? FreeCAD is always interested in attracting more developers to contribute to the software. The majority of the FreeCAD workbenches are written in Python. However, there are also some workbenches and areas of FreeCAD written in C++. There are many areas to look at, or perhaps you have your own idea for a tool that you'd like to create for FreeCAD, or an issue that you think you can solve.

The first port of call is the forum, and in particular 'Developers corner', where it would be a great idea

QUICK TIP

Remember that FreeCAD is made and maintained completely by volunteers. Be kind!

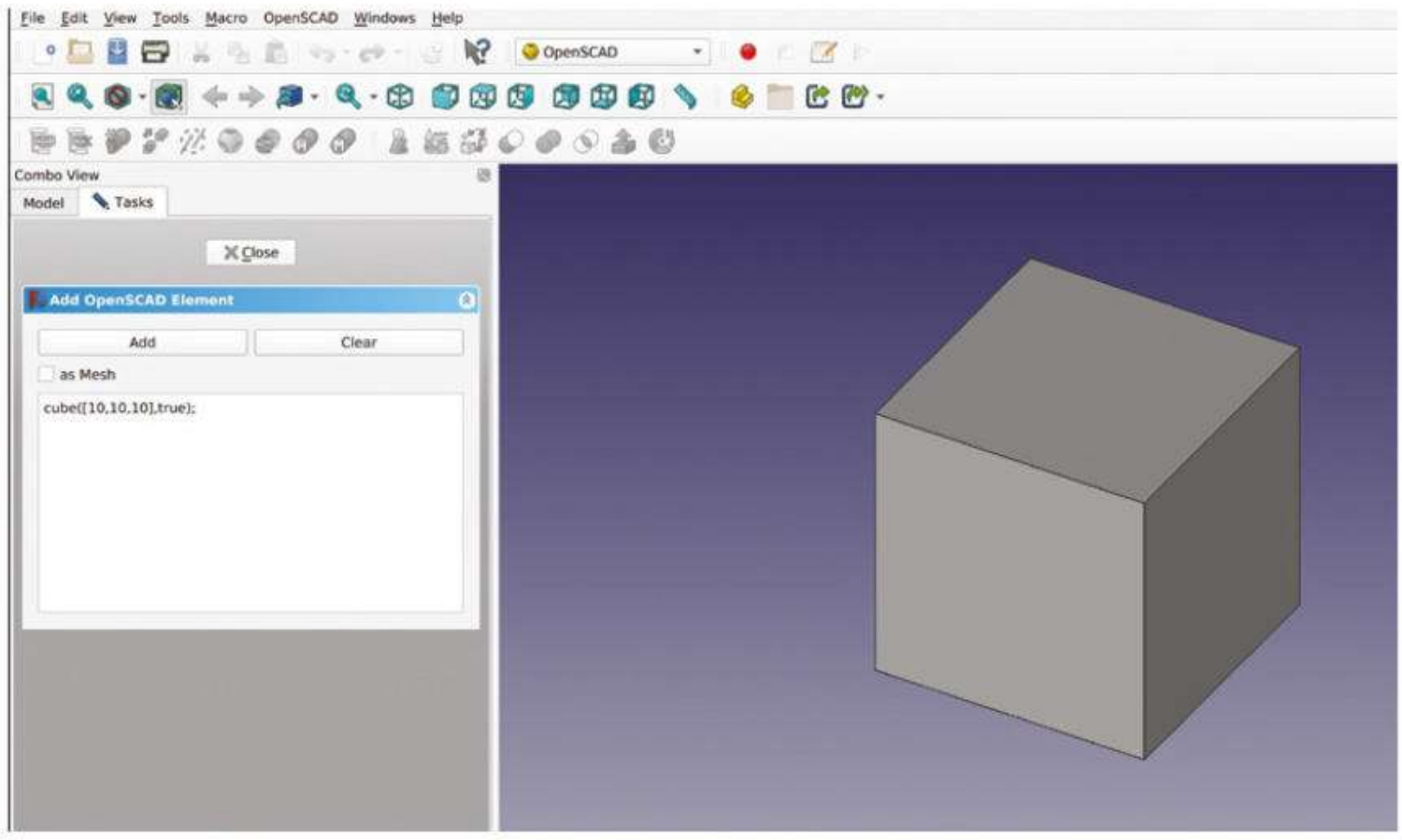


Figure 8
The OpenSCAD workbench offers all the functionality of the excellent OpenSCAD environment built into FreeCAD

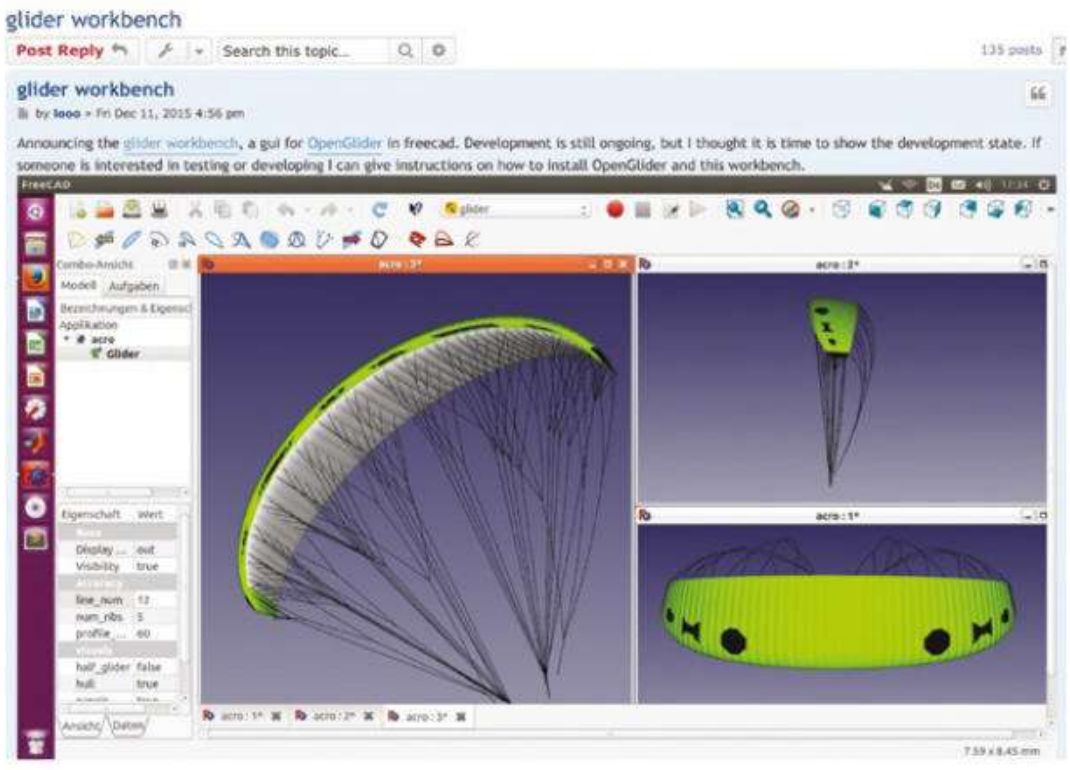


Figure 9
The Glider workbench is not under active development, but is still is being used by a small number of users to produce amazing projects

to introduce yourself. Read the pinned thread, 'Read this first if you want to write code for FreeCAD', to get the basic orientation of the FreeCAD developer community approach.

In this final part, we wanted to recap and expand upon other sources of help, should you need it, for FreeCAD. One of the best resources out there is the FreeCAD community forum – **forum.freecadweb.org.** We've mentioned it before, but it's worth repeating a couple of points about the forum. First of all, there's a really high chance that there is information that can help you with your particular situation already on the forum. It's vast, and a search query for your subject will often return a heap of golden knowledge. It's definitely worth searching before adding a new question or thread. Secondly, in order for community members to be well-equipped to answer a question, it's really useful if you post basic information about your operating system, version of FreeCAD, etc. Wonderfully, FreeCAD has this built-in: if you go to Help > About FreeCAD from any workbench in FreeCAD, you will see your system information with a handy 'Copy to clipboard' button (**Figure 10**). Click the button and paste it into your opening post, followed by your question, and it may well make it much easier for someone to answer.

Sometimes you'll find that something in FreeCAD doesn't work as you expect it to. Again, a good bit of forum searching and wiki reading will often yield a solution, but sometimes something might be broken. It's always worth, and always preferred, that you search the forum, and post on the forum, checking if others have had your experience and allowing other community members to help identify if it is indeed a bug. If the consensus is that it is a bug, then someone in the community will raise it as an issue in the correct way or will inform back in the thread that the issue is a previously identified issue. Often, the answer will be that an issue in the current stable release version of FreeCAD has been addressed in the developmental version of FreeCAD. On that subject, the developmental versions of FreeCAD are always available for people to try. We'd suggest, as we have throughout these articles, that using the latest stable version whilst you are learning is recommended, but as you understand more it can be fun to try the developmental versions, which often

Figure 10
The 'About FreeCAD' tab allows you to copy details of your machine, FreeCAD version, and operating system

show a sneak preview of new, interesting features and developments. Also, as FreeCAD is open source, there are some forked alternate versions of FreeCAD – this helps to keep FreeCAD development innovative, as the forked versions often explore different features and approaches. One popular forked version of FreeCAD, that contributes a lot of development back into the main FreeCAD branch, is the LinkStage3 fork by Realthunder – with some of the basics under your belt, you might like to try it out.

So, as we come to the end of this series, it's perhaps worth considering where to look for other sources of information and tutorials on FreeCAD, as well as perhaps places to find some inspiration! First up for inspiration, new releases, and new feature announcements – and also a great place to see challenging, interesting work done in FreeCAD – is following the official **@FreeCADNews** account on Twitter. Watching that Twitter account, you'll see the wide range of approaches people use with FreeCAD and also come across people doing CAD challenges using FreeCAD and more. They often post using the hashtag #madewithfreecad, and it's astonishing to see what people have made. YouTube is absolutely crammed with tutorial videos, and pretty much every area of FreeCAD is covered. One particular YouTube channel, Joko Engineering, deserves particular mention for excellent tutorials that not only cover using FreeCAD tools, but also often explore underlying CAD concepts.

> **YouTube is absolutely crammed with tutorial videos,** and pretty much any area of FreeCAD is covered

We've mentioned it a few times, but the official FreeCAD documentation is in the form of a wiki and is available at **wiki.freecad.org/Getting_started**. All the built-in workbenches and tools are documented, and there are numerous short tutorials on different workbenches. It's got heaps of information clearly laid out, and the wiki is already available in numerous languages. It also has wider reading on FreeCAD. A nice wiki entry that gives a quick-read version of FreeCAD's origins and history can be found at **wiki.freecadweb.org/History**.

We hope this tutorial series has been useful, and we feel that if you worked through all 16 parts, you would be pretty competent at modelling in FreeCAD. In our opinion, it's definitely a process that requires practice, and with each new model you'll find some area that requires a new approach to be explored and learnt. As such, it's a great idea to work with FreeCAD as much as possible. One thing we have seen a lot of in the communications around this series is that often CAD communities have competitions or ongoing challenges which can not only be good fun, but also provide opportunities for you to expand and learn more approaches.

We've really enjoyed writing this longer series, which has been a new approach for HackSpace magazine, and we hope that you have enjoyed it. Many thanks to those of you that have worked through all the tutorials, reaching out to us on social media to share your progress. We continue to look forward to see what you can create using the brilliant FreeCAD platform.

AND NOW...**THE BOOK**

We are really pleased to announce that Raspberry Pi Press is collating this 16-part FreeCAD series into a free-to-download book. Having all the articles in one place makes it a little quicker and simpler to work through the series, and means you can jump back to a previous chapter to check something more easily than, perhaps, digging through the pile of magazines! It is slightly rewritten, as a few small things changed in FreeCAD since the early articles were written. We hope that it will be really useful to everyone trying to learn FreeCAD.

We're putting the finishing touches on it as we go to press and it should be available to download from our website shortly.

TUTORIAL

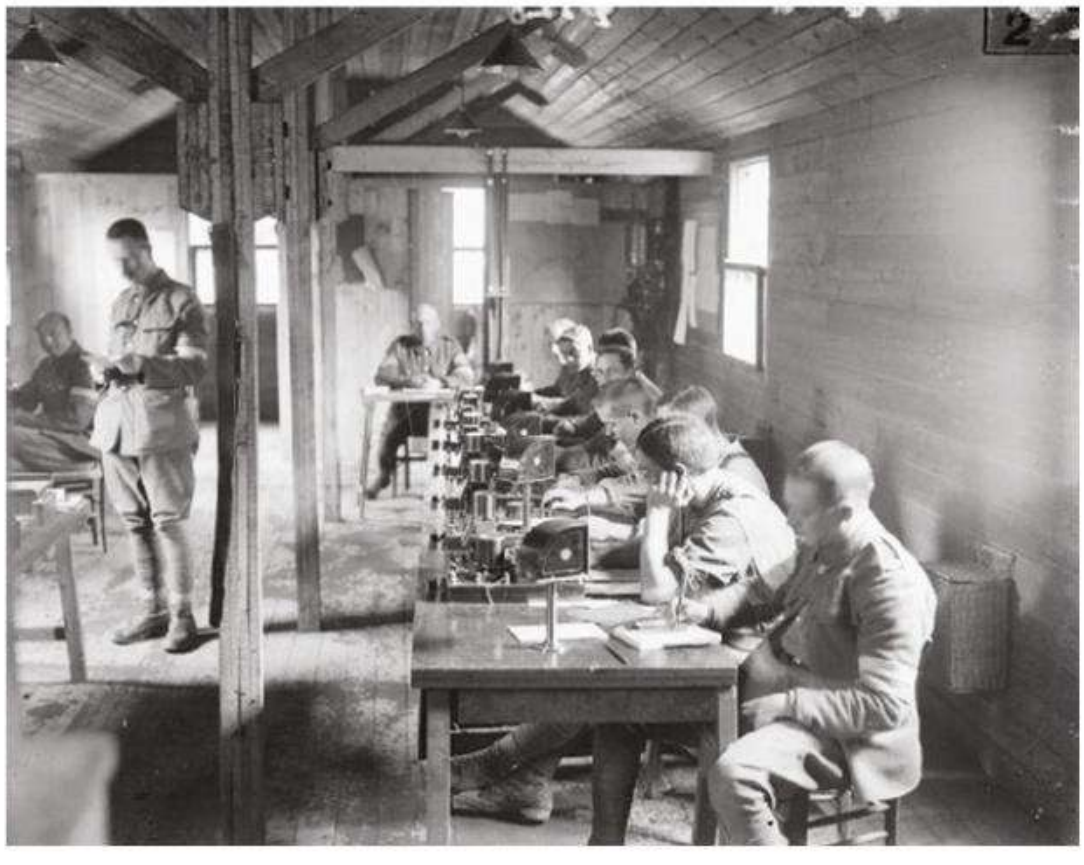
Hands on with Morse code

Mike Bedford reveals something of the first-ever code for data transmission, and shows you how to try it out for yourself

Mike Bedford

Despite loving all things digital, Mike admits to being a bit of a Luddite, vinyl records and all.

On 24 May 1844, a message was sent from Washington DC to Baltimore. Normally it would have taken a day or more for a dispatch rider to cover the 66 km, but this message was received almost immediately. The words of the message – What hath God wrought – conveyed something of the enormity of the achievement and the awe of those observing it. The secret of this momentous event is revealed by the person sending the message – Samuel Morse. Indeed, the code that bears his name was the world's first code for data transmission. Here we delve into this historic code, and discover that it was a lot more sophisticated than most people think. We'll also see that, despite it being more than a little long in the tooth, you can still hear it in use today. And if you want to get some practical experience, we'll show you how to learn Morse code, and even try your hand at sending it.



SPOTLIGHT ON THE CODE

Each letter, figure, or symbol in Morse code is represented by short and long elements – which could be sent by telegraph, radio, light, or even audibly – that are shown in written form as dots and dashes, as illustrated in **Figure 1**.

If you fancy learning Morse code, here's what we suggest. First, you need to memorise the code – it's not as difficult as you might expect. What's more, if you use Morse code to any significant degree, you'll probably never forget it. Previous generations might have learned it from a printed table, but there's a better way. Because you'll eventually want to receive Morse code by ear, it makes sense to learn it audibly, and there is no shortage of software and online utilities to help. You might be eager to try sending Morse code, but it's better to learn to receive it first. By listening to it, you'll learn the characters as rhythms, so it'll become second nature to reproduce the correct timings when you start to send it. So, you'll intuitively send dashes that are three times longer than dots, you'll send the spaces between dots and dashes in a character that are the same length as dots, and so on. It's also a good idea to learn to receive Morse code at a reasonably high speed – 15 words per minute or perhaps more – even if there are long gaps between the letters. Again, this helps you to learn the rhythms rather than trying to count dots

Left
Morse code had already been in use for 72 years when this photo of a British Army signal exchange in the Battle of the Somme was taken, and it would be used on board ships for another 83 years

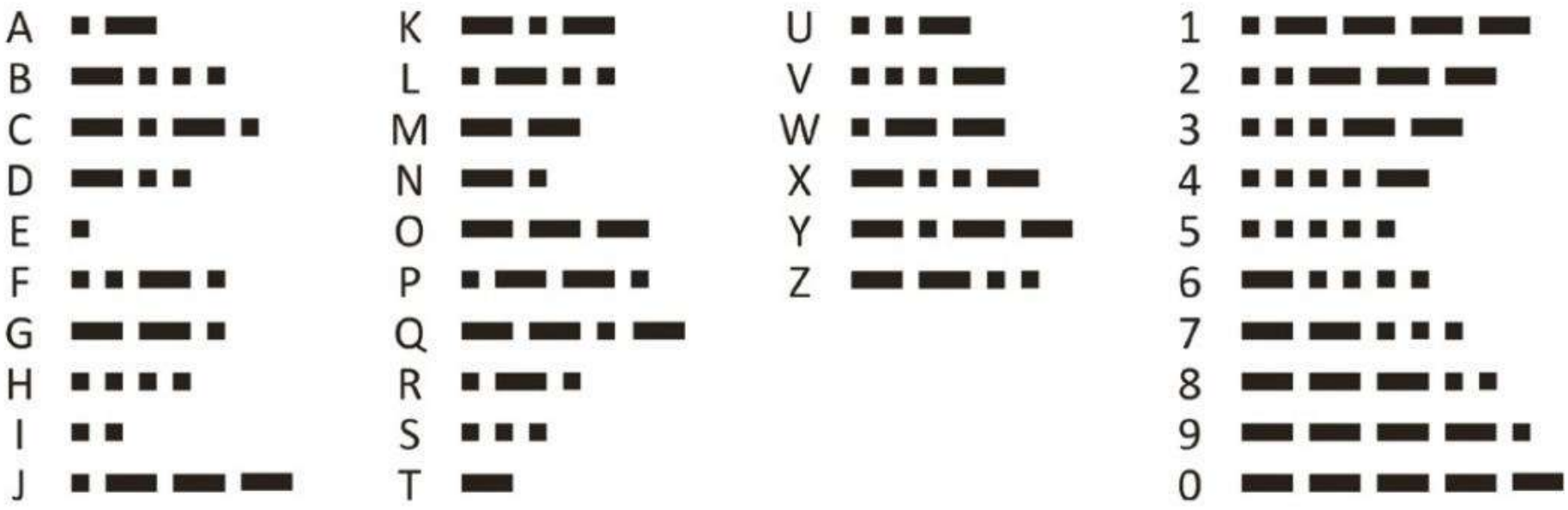


Figure 1 Characters in Morse code are represented by dots and dashes, but it's much more sophisticated than the apparently random codes might suggest

and dashes. A well-respected site for learning Morse code can be found at **lcwo.net**. It's free, although you do need to register.

When you progress to learning to send, you need a Morse key, and you can buy them new (from about £20) or second-hand ex-military (from less than £10 on eBay). Electrically, Morse keys are just like momentary action push-buttons, but they're finely balanced, and adjustable, so operators could use them all day long, even at high speed, without getting wrist injuries. Make sure you get one that works by pressing it down, not side to side (we'll look at those later). At a pinch, to start, you could even make your own key using a springy steel strip instead of a delicately engineered mechanism. You'll also need something to make a bleep, and the simplest solution is an active piezo sounder, which is the type that generates a bleep whenever a DC voltage is applied to it. All you need to do is wire the Morse key, sounder, and a battery in a series loop. If it's loud, and your family don't share your enthusiasm with incessant bleeping, you could put a piece of tape over the hole. A better solution would be to knock up a simple audio oscillator, ideally with a volume control, that can drive headphones. Alternatively, you could write some simple code on a Raspberry Pi or similar.

A SOPHISTICATED CODE

You might think of Morse code as being crude in the extreme, but you'd be wrong; it's remarkably sophisticated. If you look at the dot-dash combinations for the various letters, they might seem to be largely random, but they were chosen deliberately. The two most common letters in the English language are E and T, which are represented by a single dot and a single dash, respectively. On the other hand, J, Q, X, and Z are the least commonly used, and their codes are . - - - , - - . - , - . . -, and - - . ., which are some of the longest codes. Using a code in which letters are different lengths depending on how commonly they're used is very efficient compared to digital codes like ASCII, where all letters are the same length. Much the same method was reinvented in the 1950s as a method of data compression, and software that employs this Huffman code is still used today. Its principle differs from that of Morse code, only that instead of each character (or byte of data) always having the same code, the data is analysed and codes are assigned on the fly, according to how commonly they're found.

> You might think of Morse code as being crude in the **extreme, but you'd be wrong; it's remarkably sophisticated**

AUTOMATIC KEYS

Morse keys, like the ones we've already seen, limit the speed at which most people can easily send Morse code. Automatic keys allow Morse code to be sent more quickly, and they also force the user to send characters with accurate timing, for example, dashes always being exactly three times as long as dots. Automatic keys are operated by moving a lever – usually called a paddle – from side to side, instead of up and down.

The first automatic keys – or, strictly speaking, semi-automatic keys – were called bug keys and were purely mechanical. If you moved the paddle to the left it made an electrical connection, allowing you to send dashes, even though they had to be formed

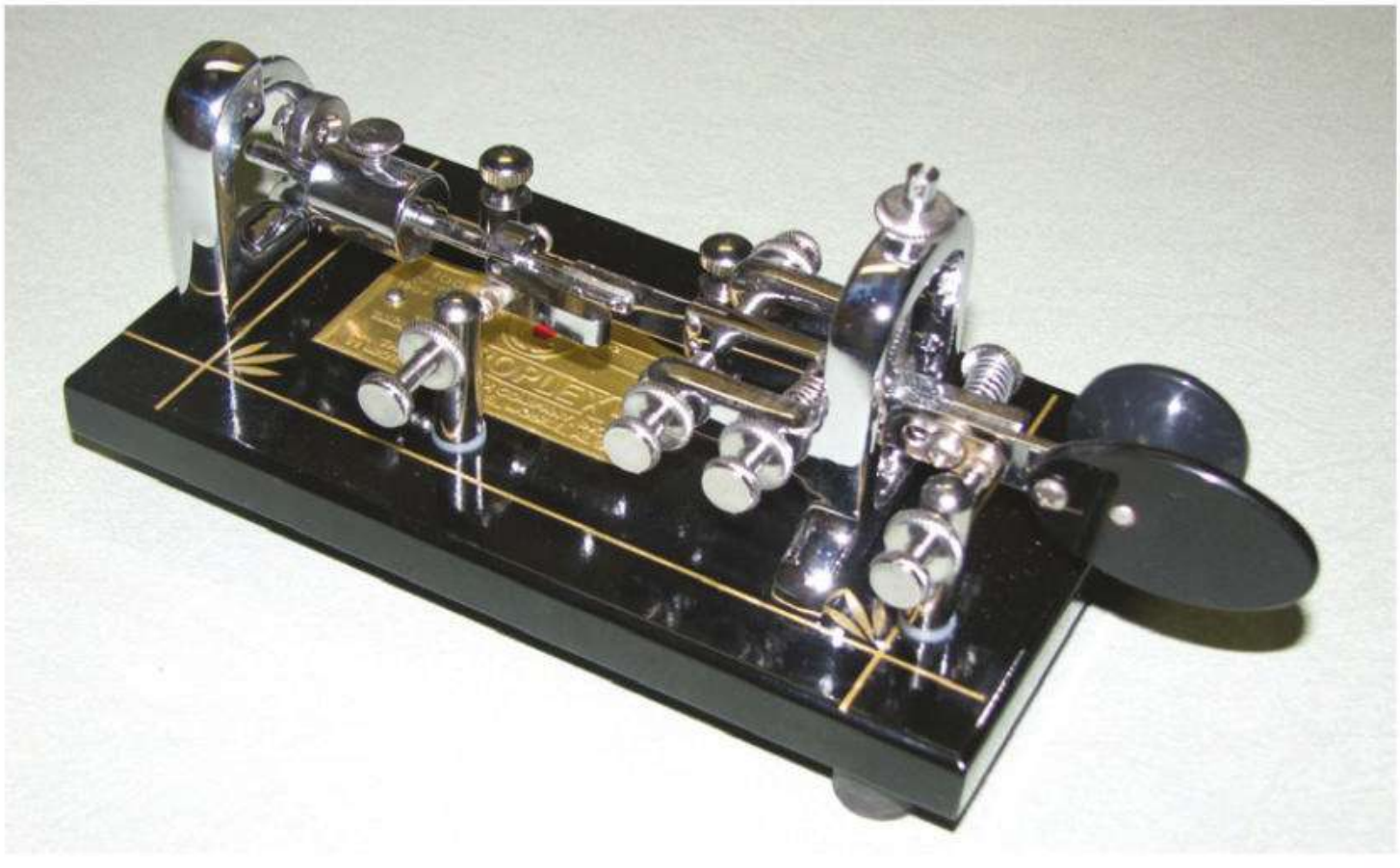
TUTORIAL

Above
You can practice sending Morse code using nothing more sophisticated than an old-fashioned Morse key, a battery, and an active piezoelectric buzzer

manually, just as they are with a manual key. However, when you moved the paddle to the right, a horizontal pendulum mechanism caused a string of dots to be sent. So, to send a letter B (- . . .), you'd move the paddle to the left for the correct length of time and then move it to the right until three dots had been sent. OK, so I chose a good example, but averaged over the alphabet, using a bug key certainly reduces the number of key movements required.

Going beyond the bug key, using mechanics is tricky, but with electronics, it's easy to produce a fully automatic key that sends strings of dashes when you move the paddle to the left and strings of dots when you move it the other way. It reduces key movements further, and it means that dashes will be perfectly formed as well as dots. If you'd like to try this way of sending Morse code – but, as it's generally suggested, not until you've mastered sending it manually – you'll either have to buy or make a side-to-side key that makes a different circuit depending on which way you move it, and build the simple circuit that we show in **Figure 2**, which is based on a design by Indian radio amateur Hari Sankar (call sign VU3NSH). It might seem strange that we've mixed logical families, using both a 4000 series and a 74HC series chip, but that's just because we had a 74HC32 at hand. It'll work just as well with a 4071 instead of a 74HC32, although note that the pinout is different. You can use pretty much any type of diode and any bipolar NPN transistor.

And it gets better. Some side-to-side keys have two paddles instead of one, so any electronics connected to the key can detect three active conditions: either of the paddles being closed by themselves, and both



paddles being closed at the same time because they're squeezed together. This type of key is usually used with more complicated electronics. It generates strings of dots or dashes, as the appropriate paddle is pressed by itself, or strings of alternating dots and dashes when the two paddles are squeezed together. So, if you squeeze the paddles together, with the dash paddle closing slightly before the dot paddle, the letter C (- . - .) could be sent. Circuits using discrete logic chips abound, but I'd be inclined to suggest you use something like a Raspberry Pi Zero. As well as implementing strings of dots and dashes, and alternating strings, you could also provide automatic spacing. Of course, short spaces the same length as →

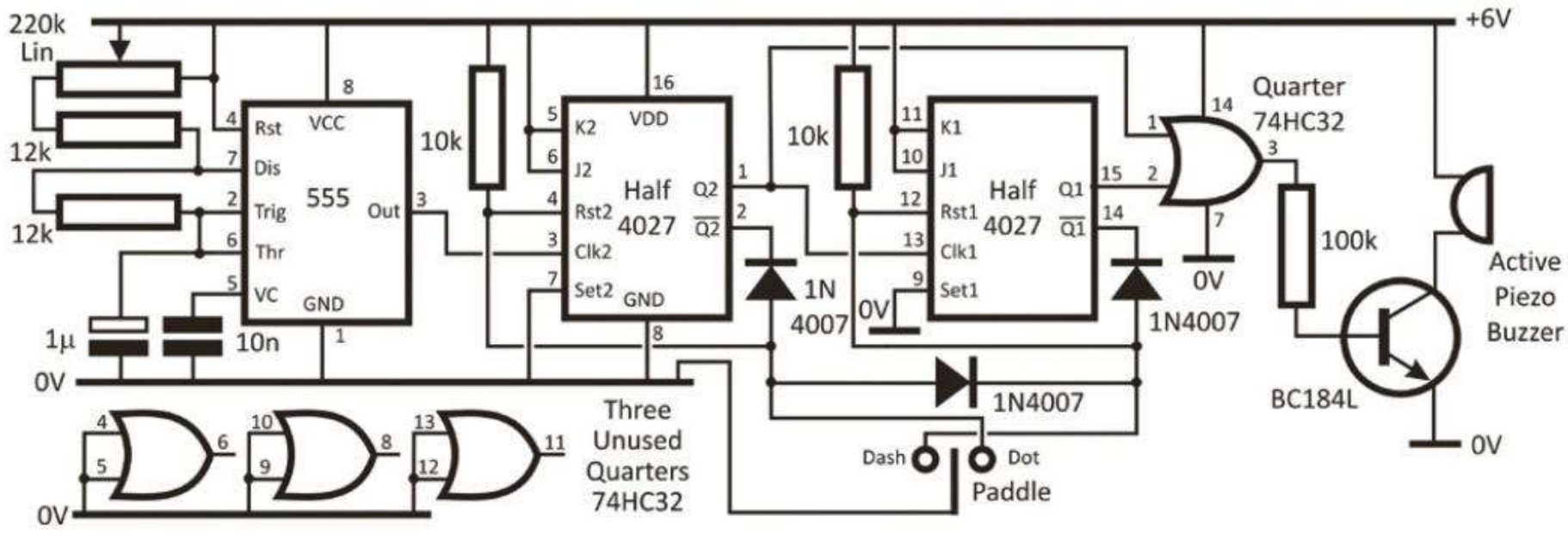
LISTEN **TO MORSE CODE**

If you want to hear Morse code on the air waves, and if you have a shortwave receiver, you'll find it at the lower frequency end of each of the amateur radio bands, although which band you should choose will depend on various things, including the time of day. Not having a suitable radio is no handicap thanks to the network of WebSDRs – online software defined radios. Go to **websdr.org** and take your pick. We suggest choosing one that covers the 40 m amateur band (7.0 to 7.2 or 7.3MHz, depending on the country) and listen in from just above 7.0MHz to about 7.06MHz – at higher frequencies you'll find voice signals.

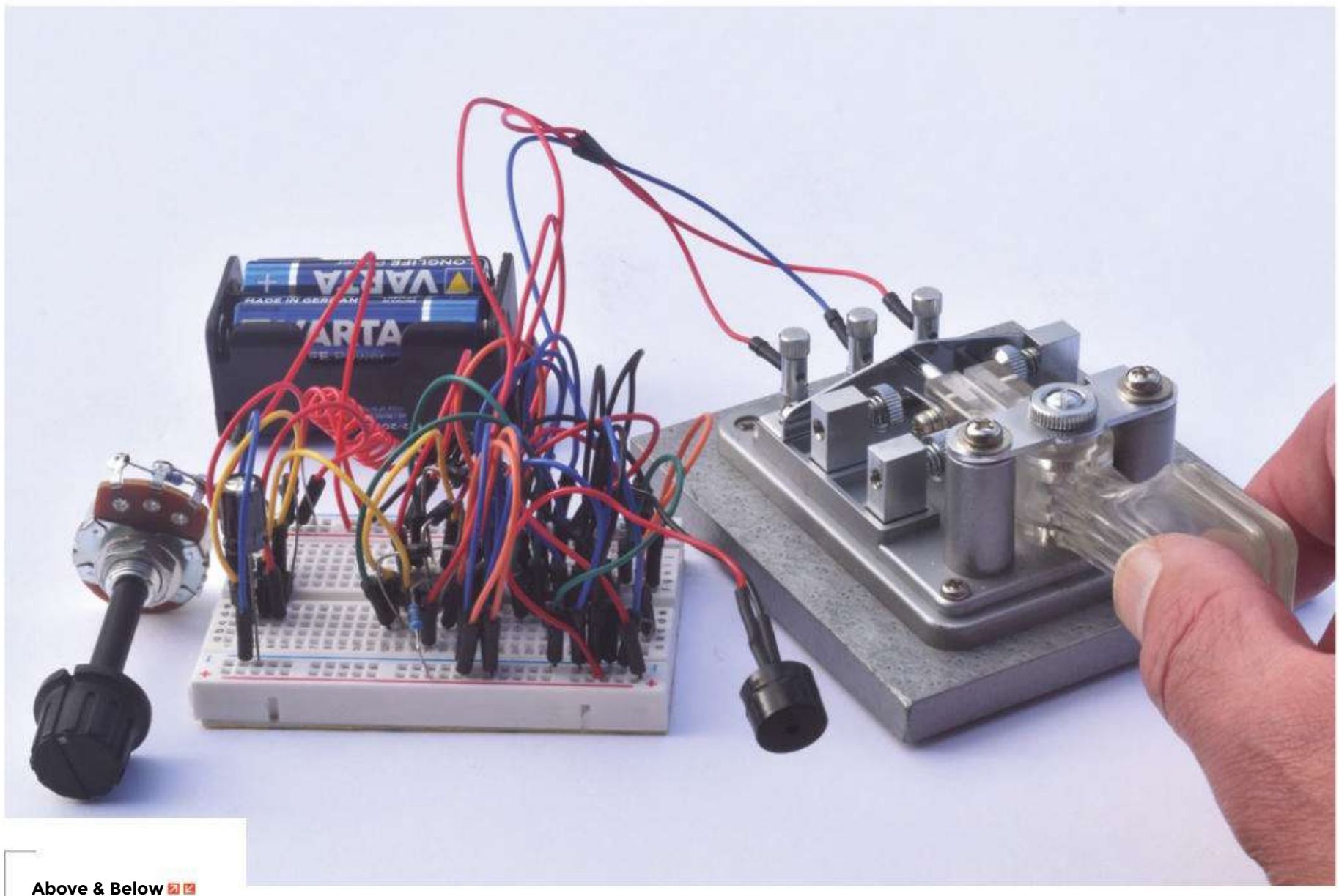
Left
Strings of dots can be sent with a semi-automatic 'bug' key, like this 100th-anniversary edition of the Vibroplex, the first Morse key ever to provide a degree of automation

Image: CQDX.ru

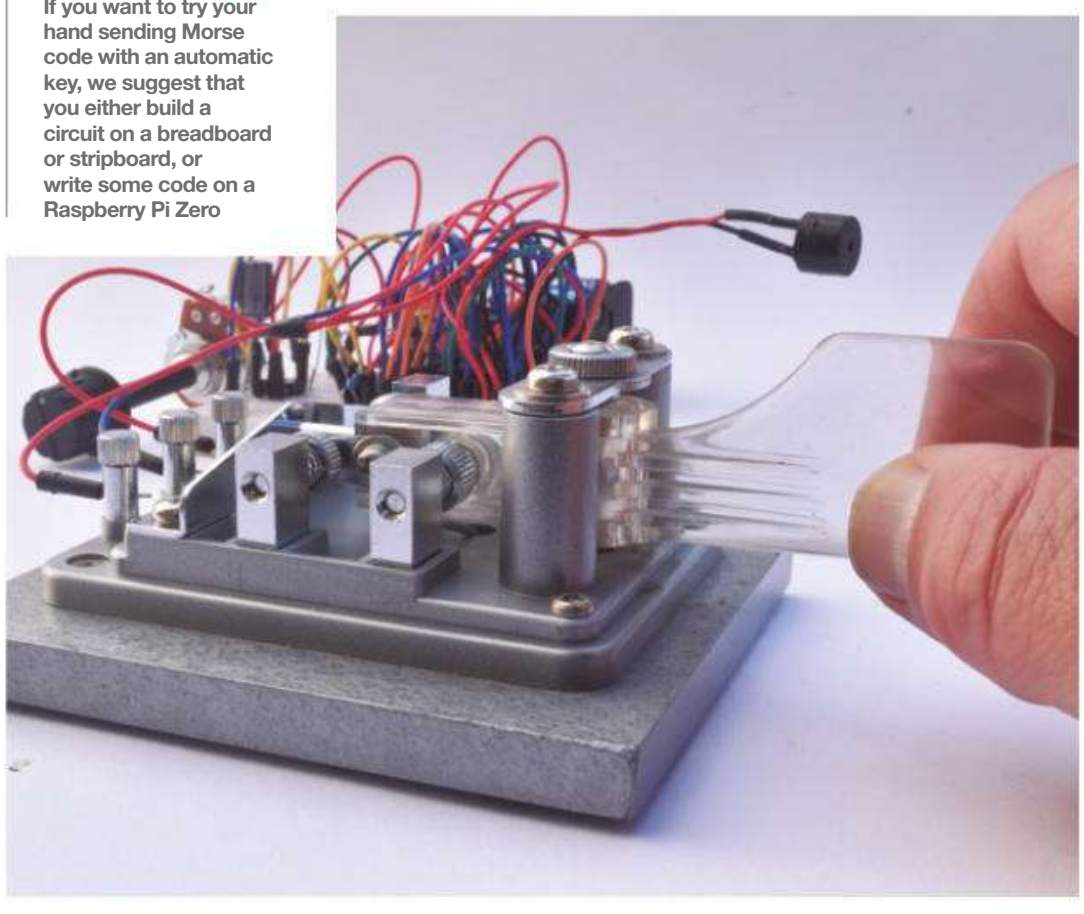
Figure 2
This circuit can be used with a side-to-side key to implement an automatic key that takes the donkey-work out of sending strings of dots and dashes



TUTORIAL

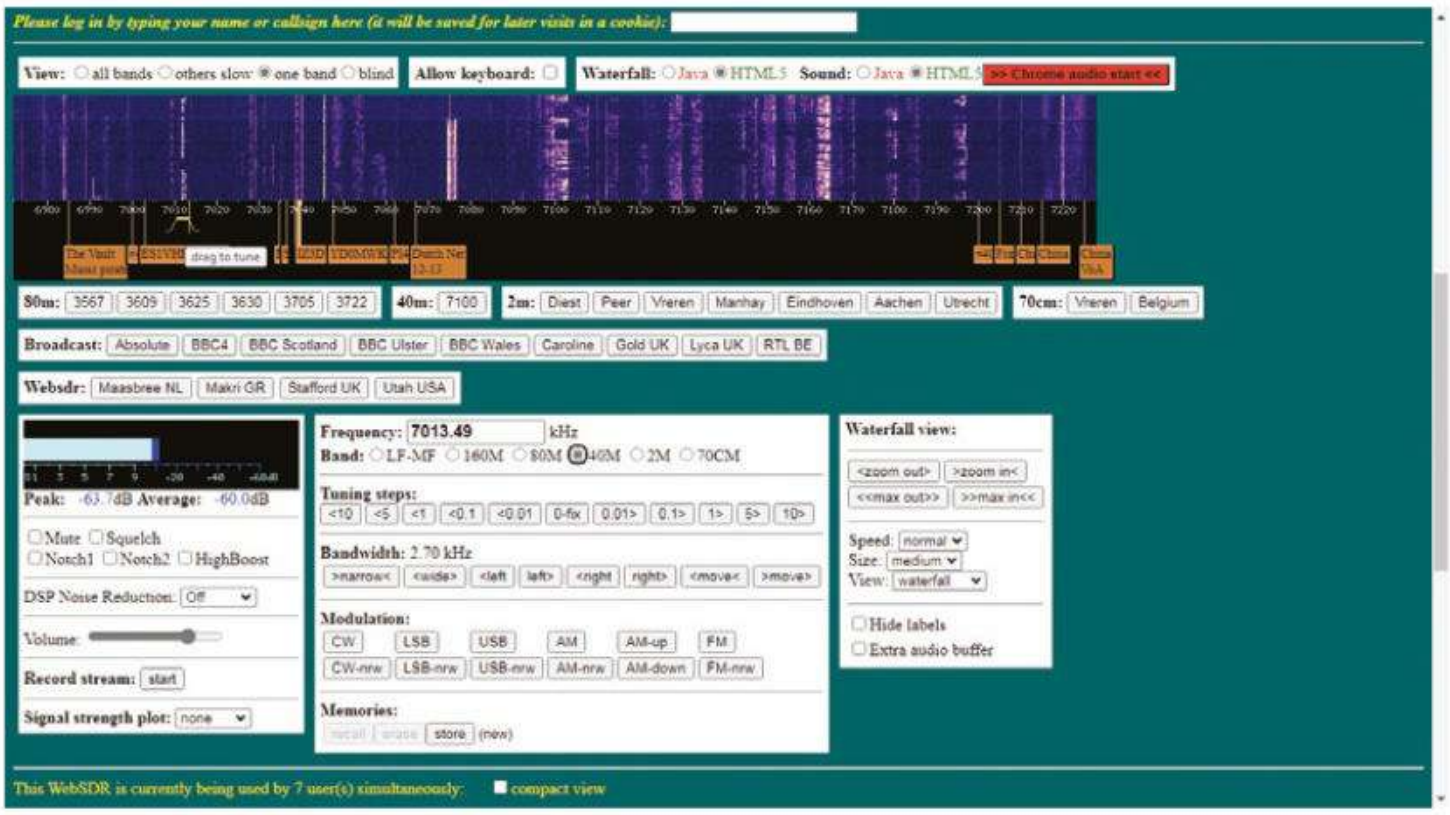


Above & Below
If you want to try your hand sending Morse code with an automatic key, we suggest that you either build a circuit on a breadboard or stripboard, or write some code on a Raspberry Pi Zero



dots will be an integral part of strings of dots and/or dashes, but if the paddles aren't pressed for, say, two dot lengths, you could prevent any more strings being sent until three dot lengths had elapsed, this being the length of a space between individual characters. So, if you try to send each character slightly early, the spacing will always be perfect. Similarly, you could enforce the space between words of exactly seven dot lengths.

Of course, if you're going to implement a squeeze key on an SBC, if you have a keyboard attached you could also create a fully automatic keyboard sender. These have been used but, surprisingly, you might think, they're not as popular with radio amateurs as side-to-side keys. Partially this is because many radio amateurs like to hone their skills in sending Morse code, skills that are barely needed with a keyboard sender. However, there's a more practical reason. Because software for reading Morse code often isn't as good as reading it by ear on noisy shortwave bands, it's usually read by ear. But, since most people can send Morse code more quickly than they can



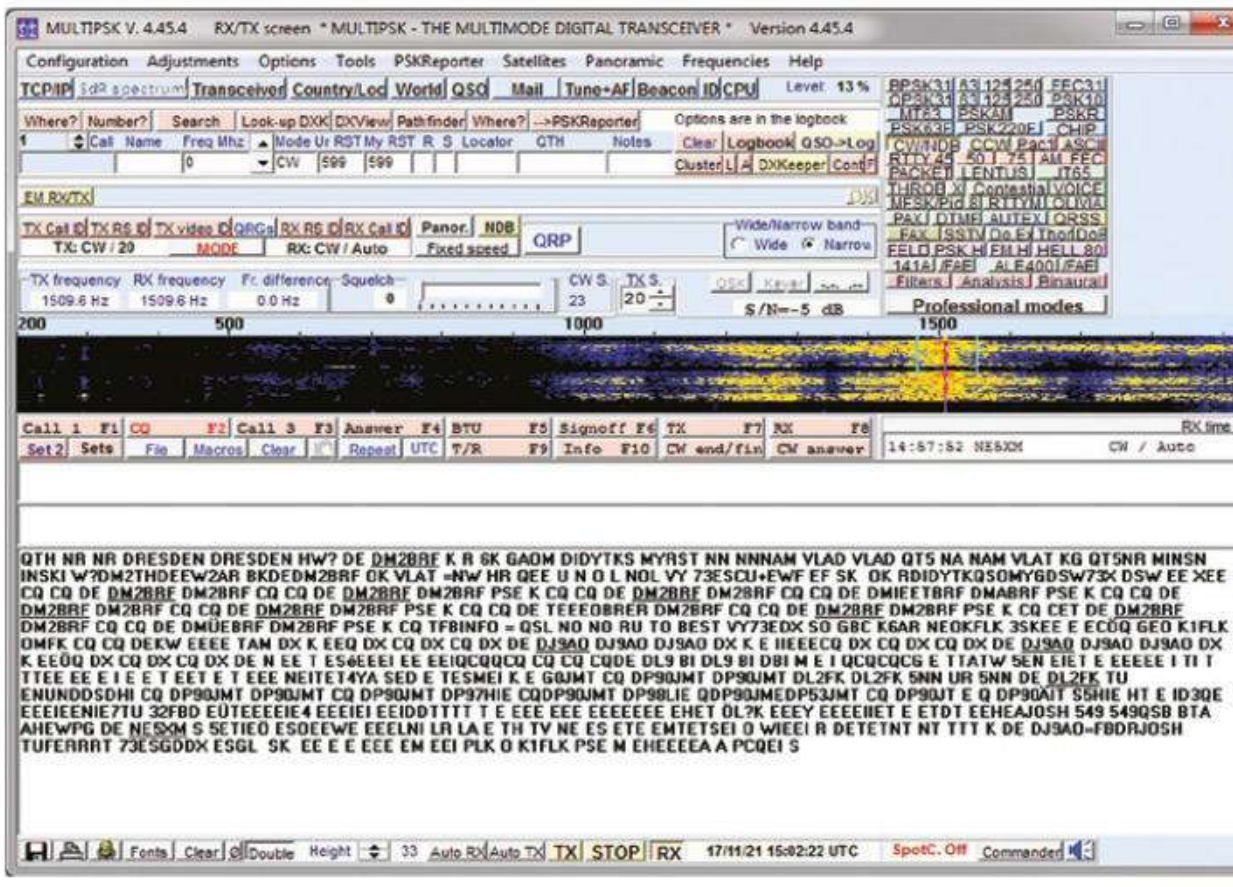
Left
Software like MultiPSK can decode signals on the amateur bands, but it will still take some time to understand the abbreviated nature of Morse code contacts

Below
Most of us don't own shortwave receivers, but we can receive Morse code on the amateur bands using the global network of WebSDRs

read it, even using a manual or side-to-side key, the benefit of sending very fast Morse code using a keyboard would be a retrograde step.

MORSE CODE TODAY

Morse code was phased out for maritime communication at the end of the 20th century, but that doesn't mean it's dead and gone. Today it's used by radio amateurs, and it spans longer distances than speech, all other things being equal. Using a simple, homemade Morse code transmitter, radio amateurs are able to send signals anywhere in the world. When most of us now carry a radio communication device that allows us to talk easily to anywhere in the world, this might not seem like a big deal, but we're not comparing like with like. Our mobile phones communicate with a base station usually no more than a few kilometres away, while simple transmitters and Morse code have covered almost 20,000 km. Not bad going for a method of communication that first saw the light of day almost 178 years ago.



DECODE MORSE AUTOMATICALLY

If you're new to Morse code, you're really going to struggle to interpret any code you might hear on the amateur bands. Software can lend a hand, though. It won't do a perfect job, because hand-sent Morse code isn't perfect, and the shortwave bands contain lots of interference, but if you pick a loud signal, you should have some success. Our recommended decoding package is MultiPSK, which is free for Windows from **f6cte.free.fr**, or fldigi (**w1hkj.com**) for Linux. You'll need to route the audio output from your browser to the decoding software while you're listening to the WebSDR. Possible utilities are the JACK Audio Connection Kit for Linux, or VB-Audio Cable for Windows.

TUTORIAL

Moving beyond the K40 part 2

Building the gantry

Dr Andrew Lewis

Dr Andrew Lewis is a specialist fabricator and maker, and is the owner of the Andrew Lewis Workshop.

In the second part of this series, you'll be looking at building a gantry for a homemade laser cutter, and the components that you'll need to make it move under computer control. In the last part of this series we took a general look at the anatomy of a laser cutter, and considered the outer case. In this instalment, you'll be looking at the gantry and bed of the laser cutter, making a robot that moves in the X, Y, and Z axis. If you've looked closely at the K40 laser, you've probably noticed that the gantry looks similar to an old MakerBot Replicator. One stepper motor uses a timing belt and pulley to move a rail along the X axis, which supports a second motor moving the lens mount in the Y axis. You will be putting together a more robust version of this system, with components better suited to the larger scale of the system you're building. Exact dimensions will vary depending on the choices you made when you built the enclosure, but the principles behind the design will be the same for most laser cutters.

RIDE THE RAILS

The gantry of a laser cutter is designed more like a 3D printer than a CNC router. It doesn't have to move a lot of weight around, and the effects of tool pressure won't be an issue unless you're using the planet-killing laser from the Death Star. The gantry you're building should be free moving with a light platform, and able to move as quickly as possible for operations like raster engraving. 3D printing can

Above
This sled is designed to hold the laser cutter's mirror and lens assembly on 12mm linear rods. You can see the linear bearings haven't been properly installed yet, because there are no circlips on the ends of the bearings, and the mountings have no bolts through the holes to stop them from flexing. The whole assembly is lighter than a 3D printer head, and can be positioned easily with stepper motors

help here, with so many pre-existing 3D printable designs to choose from. You just have to choose the appropriate rails, bearings, and motors for your particular needs.

The Z axis of the machine is a tricky proposition. The stock K40 machine has no adjustable focus or bed height, which means that if you're using a thicker material that's outside the laser's focusing sweet spot, you're going to get very poor performance. Many K40 users build a manual or electronic adjustable bed, or use scissor jacks to position parts inside the machine. Scissor jacks don't work particularly well on larger machines, so vertical lifts at each end of the bed are a better option. There are 3D models available for Z axis lifts (and other parts) that use NEMA 23 motors with 12 mm rails in the online resources for this article.

QUICK TIP

The control board and software for the K40 doesn't support focus or bed motion. To move the Z axis of the machine under CNC control, you'll need a new control board.

STAYING SAFE

A laser cutter is a dangerous machine. Invisible laser radiation can disfigure or blind you permanently. Stepper motors can be surprisingly fast, and power mechanisms are capable of pinching and slicing through meat and bone. Remember that you are made of meat and bone. Always wear laser safety goggles when working with the laser, and be aware of the risks from mechanical parts, high voltages, and reflective surfaces.

Above
Sketches aren't enough. Dry fit everything before you commit to final positions. It's very easy to miss something and find that moving parts interfere with each other. You can see from this photo that the clearance between the parts is only a few millimetres

TUTORIAL

Right
The Z axis assemblies bolt to each side of the case, and hold the machine bed at each end. Notably missing from this photograph is the end-stop, which would be positioned near the motor so that the bed of the machine moves away from the lens assembly when the machine resets. This minimises the chance of the head crashing into the object on the bed in the event of a power failure

WHICH RAIL?

There are three main types of rail to consider when designing your laser cutter. The first and most obvious is the bright steel bar. It's common to see 8mm bright steel bar inside desktop 2D and 3D printers. It's cheap and easy to work with, but has a couple of limitations. Longer lengths of bar can sag, and so larger diameter material is needed as you increase the distance that you're trying to cover. The bar is also heavy, which can be a problem if you're using it to build a moving rail for your Y axis. If the weight is going to be an issue, you could consider using surface ground tube instead of solid bar, although it may be more expensive. Bar stock relies on linear bearings or bushings to achieve smooth motion.

Round rail is an option for longer spans, where rail is impractical. Rail is essentially round bar with a support fixed to the bottom to allow fitting onto flat surfaces. The extra support is heavy, but prevents sagging. This makes it a good choice for the longer X axis of the laser cutter, but terrible for moving rails (because of the extra inertia and momentum). Round rails use a split linear bearing to achieve smooth motion while still allowing clearance for fixings.

Adjustable linear guide rails are the premium choice for accuracy and durability. The rails are precision ground from high-quality steel, and can handle high compression and variable loads without any problems. They can be shimmed and adjusted to get very high levels of accuracy. They are expensive, high-quality products, and they vastly exceed the tolerance requirements for a laser cutter. If you have access to them and are looking for the best possible accuracy, then they are a suitable choice. In 99.99% of cases, they're complete overkill and won't give you a noticeable increase in performance over standard round rails or bars.

The mechanical parts of the gantry are pretty simple to assemble if you've ever put together a 3D printer, but what about the electronics? Unlike a 3D printer, there are some special considerations that come into play with a large laser cutter. While the K40 uses a single supply to power the laser and motors, this isn't advisable on a larger machine.

> The mechanical parts of the gantry are pretty simple to **assemble if you've ever put together a 3D printer**

In fact, it isn't advisable on the K40 either, but that's how things are, and you're just going to have to let the knowledge haunt you as you lay awake at 3am. Large stepper motors need more power. It's far better to split the high and low voltage power onto separate power supplies so that your laser supply only has to worry about powering the laser itself. Everything else should be split onto a supply or supplies with 5V and 24V outputs that can support more than the maximum output of all of the motors running at once.

QUICK TIP

Ensure your rails are parallel by using a bar of the exact length as a spacer between them.

QUICK TIP

Don't rely on the fuse in the mains plug to protect your system. Add a fuse-box or appropriate in-line fuses on the DC side of your power supplies to increase protection and reduce the risk of an electrical fire.

Left, Below
You can increase the accuracy of your rails and decrease the amount of noise they make by using bushings instead of roller bearings. 3D printers often use igus drylin bushings to decrease noise and improve print quality. Be aware that bushings are much less tolerant of misalignment than ball bearings, and don't need to be lubricated

An issue with larger machines is the increased possibility of interference. Motors and power supplies can generate magnetic fields large enough to mess with the control signals being sent to the motor. There are a few tricks that you can use to minimise the chance of this happening. Keep the signal and power wires separate as much as possible, and use shielded, twisted pair cable for signals. Braiding the stepper motor wires can also help dampen the effects of an unwanted magnetic field. Aviation plugs and sockets are an excellent choice to transport wires through the chassis of the machine. They're metal, screw together, and can handle a decent amount of current.

JUDGE ME BY MY SIZE, DO YOU?

For the Z axis of the machine, it's best to use powerful motors that can cope with the weight of whatever material is placed on the bed. The Z axis uses two motors, and these can either be wired into the same controller, or wired into separate controllers and linked in the controller software to act as a single unit. Most control software (e.g.

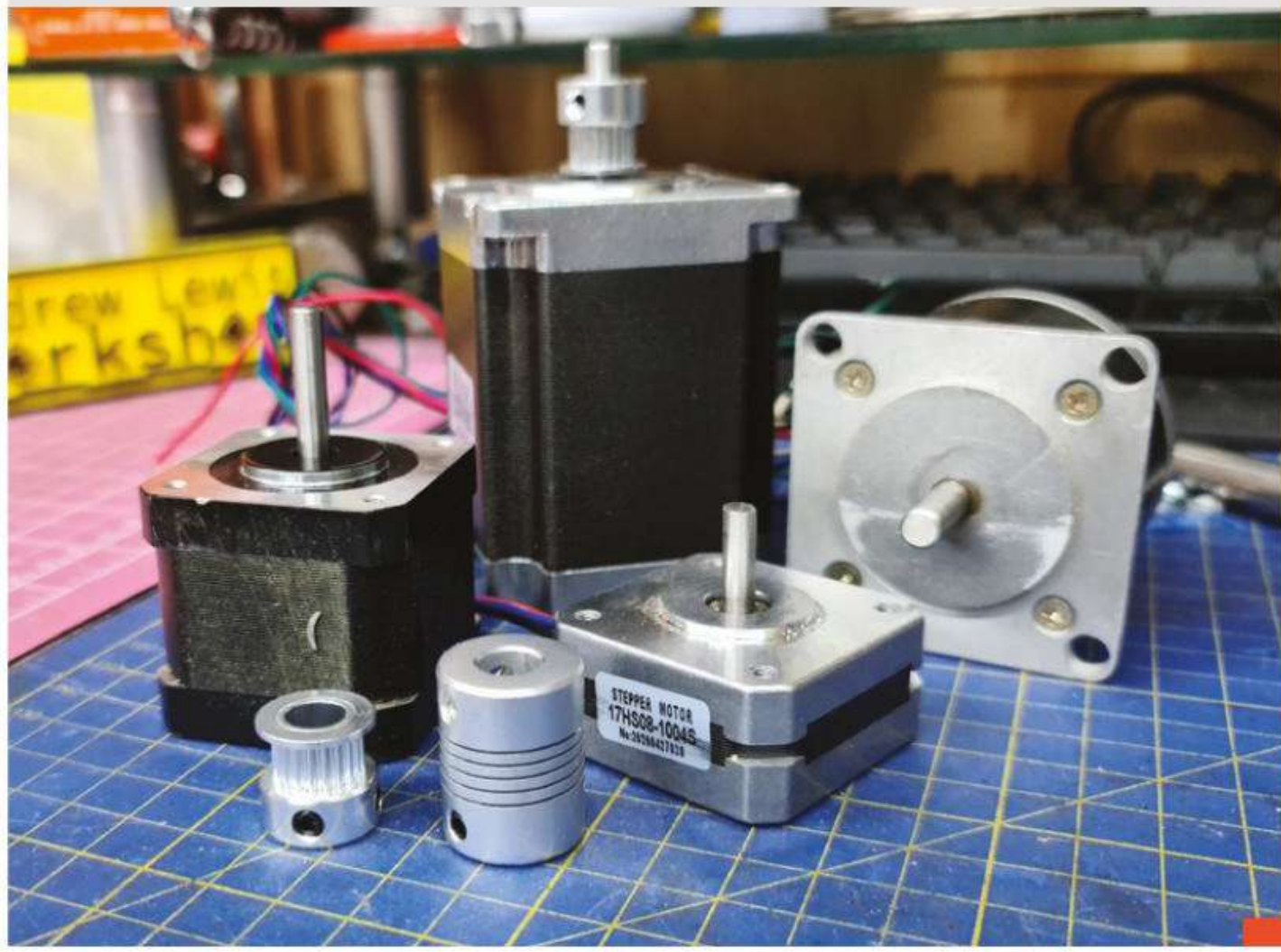
TUTORIAL

STEPPER MOTORS

Right
More industrial-style motor controllers are significantly larger than the DRV8825, but that extra surface area allows for larger cooling fins. Larger controllers are usually set by manipulating the position of a series of DIP switches to select the motor's coil current, level of microstepping, and idle behaviour from a table printed (usually) on the front of the case

Stepper motors come in a variety of shapes and sizes, but don't be fooled into thinking that bigger is necessarily better. Yes, larger stepper motors are generally more powerful, but it is possible for a NEMA 17 motor to have more usable torque than a NEMA 21. For a NEMA 17, the holding torque is generally about 3 kg-cm, for a NEMA 23 it's normally around 15 kg-cm. It all comes down to the specifications of the individual motor, and also to the controller that it's connected to. The values for individual motors might be much higher or lower than the examples given here, so always check the datasheet.

In the K40, the motor driver is built directly onto the controller board. This is a terrible idea, and means that if the motor is overloaded or shorts out, the entire board will fail and need to be replaced. Most 3D printers have removable motor drivers, taking advantage of A4988 or DRV8825 stepper controller modules. These are ideal controllers for smaller motors, requiring only a step and direction pulse from the control board to activate a stepper motor. Unfortunately, the size of the motors needed for a large laser cutter preclude the use of these controllers in most cases. For larger motors, something like the TB6600 motor driver is more appropriate. Note that many control boards are designed to expect an A4988 or DRV8825 plugged into them, and you might need to create some jumper wires to connect to a larger driver like the TB6600.



QUICK TIP

Remember that motor drivers, power supplies, and microcontrollers get hot and may need active cooling. Don't box everything in without adequate ventilation.

Below
End-stops come in a variety of shapes and sizes, and are easily available thanks to the prevalence of 3D printers in the community. Most end-stops have an indicator LED to show when they've been triggered

Above
There are a wide range of controllers available that can be used to run a laser cutter. Some are geared towards 3D printing, while others are more generalised CNC controllers. While some are designed to attach to an Arduino or other motor controller, others are self-contained units with their own microprocessor attached

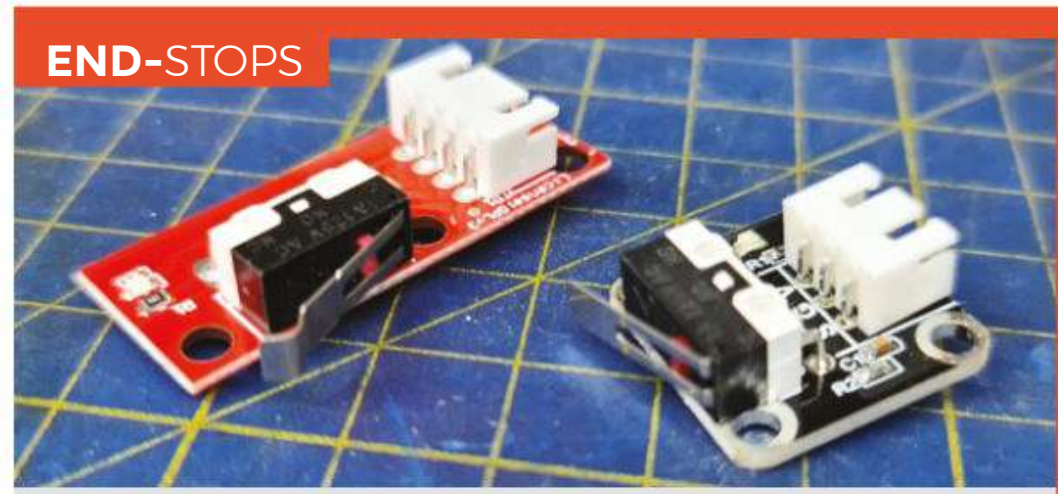
GRBL, Marlin) support dual motors on the Z axis, and there are explanations for setting this up available on their respective project sites. If you are opting to wire the motors together into a single controller, then you have the option of choosing either parallel or series wiring for each coil. Wiring in parallel is best if you have a powerful enough motor driver to handle the extra current.

The X axis motor needs to move the Y axis assembly, including the motors, mirrors, and rails. A powerful NEMA 17 should be enough for this, although a moderately powered NEMA 23 would work too. It's advisable to move both ends of the rail on the X axis, and this can be achieved by dual motors, or by a long shaft with a pulley at each end; as is seen in the original MakerBot Replicator and clones. Mounting a motor on each side looks less cluttered, but adds additional costs to the project. The Y axis motor only needs to move the sled with the mirror and lens assembly, so a NEMA 17 should be easily powerful enough for this. With motors connected and power to the controller board, you can configure the board with your choice of software. Configuration can be quite complicated, so it's worth hopping over to the website of your software choice and reading through the documentation there before you begin. If you're in doubt, grab an Arduino or an all-in-one board and try Marlin first. If you're feeling adventurous, have a look at grblHAL, which has work-in-progress support for the Raspberry Pi Pico.

QUICK TIP

Don't forget to ground the chassis of the machine! You don't want to be at risk of electrocution if a wire works loose.

END-STOPS



Every motor on the laser cutter needs at least one end-stop fitted so that the control board knows when the motor has reached the end of the rail and needs to stop. At its simplest, the end-stop is just a mounted switch. In reality, most end-stops have a small circuit board attached that do a couple of important things like debouncing switches and providing an LED indicator light when the end-stop triggers.

Optical end-stops are the best choice on a 3D printer because they have no moving parts and so won't wear out. On a laser cutter, optical end-stops can be problematic. Cutting with a laser generates smoke, and that can interfere with the operation of an optical end-stop. For that reason, mechanical end-stops might be a better choice.

THE CLUE IS IN THE NAME

End-stops can be either normally open or normally closed (or both). The safest option is to wire the switches to be normally closed so that the end-stop signal will drop low if the switch wire is damaged. That makes it less likely that a broken contact will cause the gantry head to crash, and will reduce the chances of EMI causing a false trigger.

Hardware (optical or mechanical) end-stops are usually used at the 'home' position so that the machine's origin is set by the switching of the end-stop. On many machines, the other end of the rail is protected using a software end-stop, which means that the control board knows the machine's absolute maximum size and tries to keep track of the position of the motor, stopping when it reaches what it believes to be its maximum travel. Using software end-stops is quite common on both 3D printers and the K40 series of laser cutters, although it isn't without risk. If the motor slips or loses its position, it's possible for the gantry to crash into the side without a mechanical end-stop and just keep going until something breaks. Most boards support the use of end-stops at both ends of the rail, so you can add them if you want the extra security.

TUTORIAL

Beautiful PCBs with custom footprints

Design your PCB parts in Inkscape and import them into EasyEDA

Ben Everard

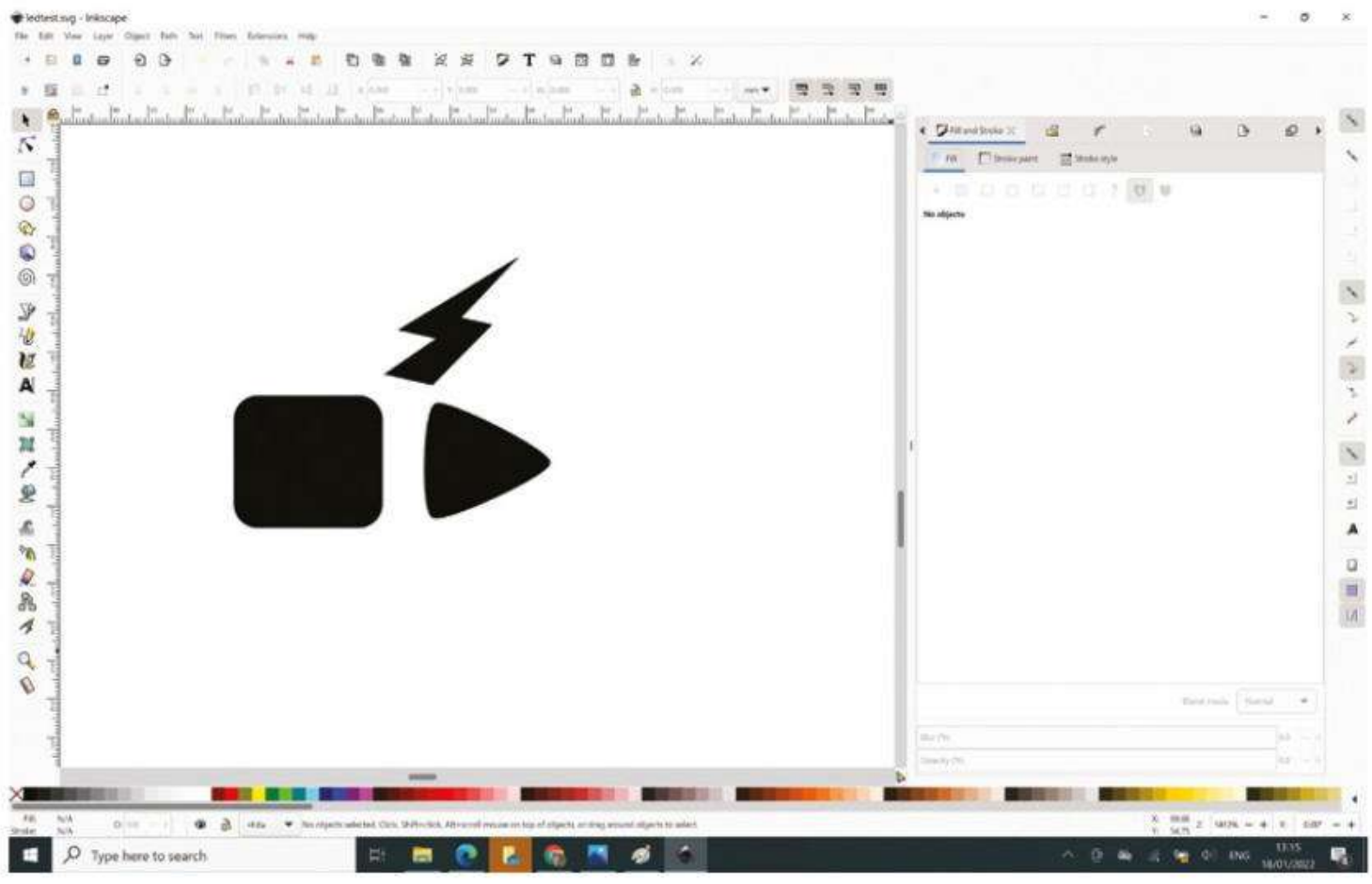
@ben_everard

Ben's house is slowly being taken over by 3D printers. He plans to solve this by printing an extension, once he gets enough printers.

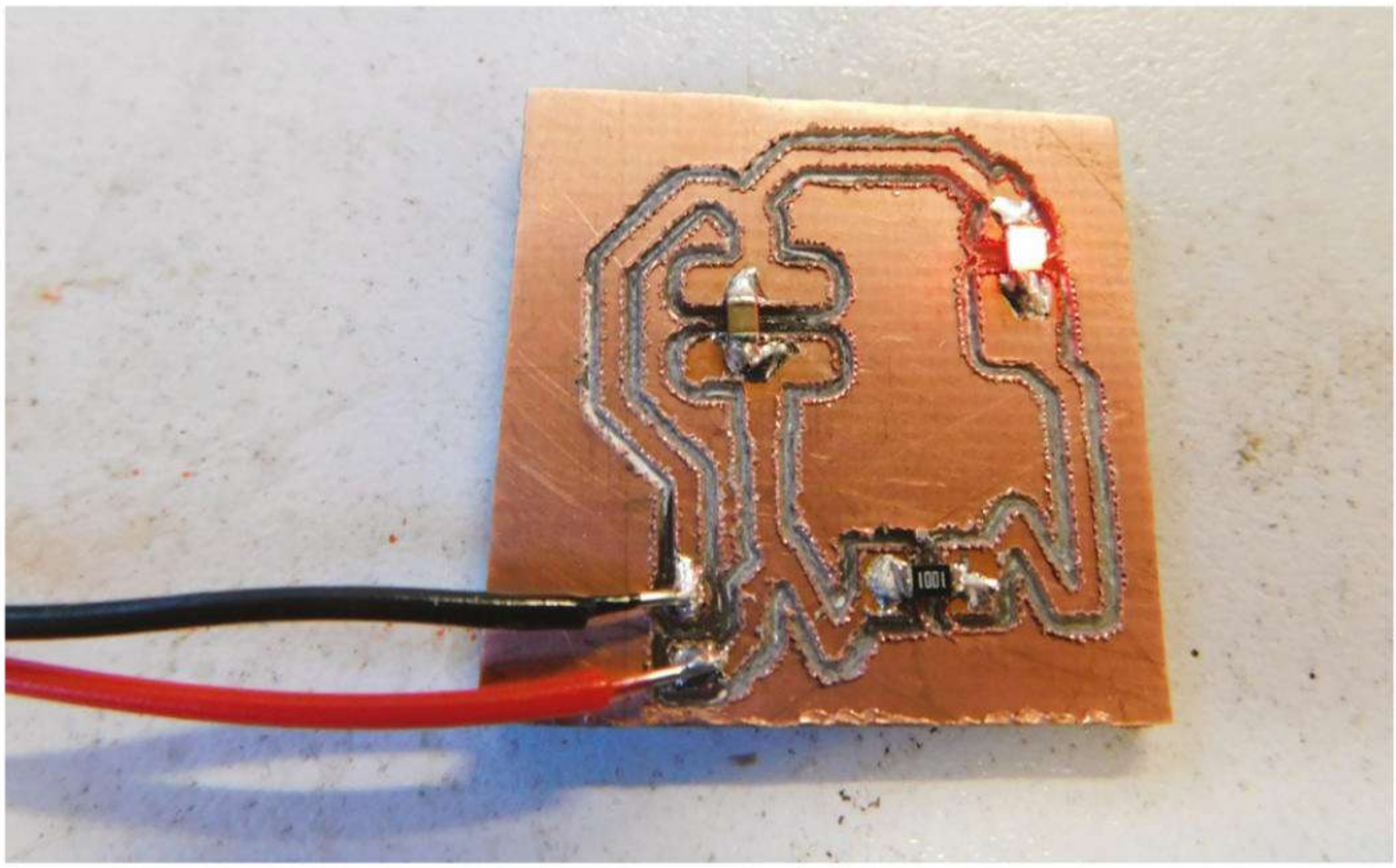
People make PCBs for many reasons. I make them mostly because I like the look. Almost all the projects I work on are simple enough that they could be done on protoboard, or even 'dead bug' style without any form of board at all. The reason I design my PCBs is because I like things to look interesting. Not neat and tidy – that's not my aesthetic – but something that makes you want to peer closer. Something that invites you in to look at it.

There are lots of ways of doing this. The choice of components, board outlines, and silkscreens all play a part. This issue, I'm going to look at something that's a bit more integral – footprints.

There's actually quite a small set of components I generally use. Resistors, capacitors, and LEDs all make regular appearances at 0805 size, and WS2812B LEDs. Around these, all sorts of other components fit in, but these four things make up the majority of almost every board I've designed. Up until now, I've always used the default component



Right
The LED footprint (sort of) looks like the circuit symbol for an LED



> There's a huge range of design programs that work with SVGs, **but Inkscape is the most common for hobbyists**

footprints, but as I use these a lot, I thought, why not create something a little more interesting? Let's take a look at how to build more elegant footprints for PCB design.

We're going to use EasyEDA because that's my PCB design tool of choice. There's a very similar workflow for KiCad.

Scalable Vector Graphics (SVG) is the most common format for vector graphics. Since our PCB also uses vector graphics, it makes sense to use this. There's a huge range of design programs that work with SVGs, but Inkscape is the most common for hobbyists. You can download it for free from **inkscape.org**. It's a complex tool and we won't be covering it in detail here, but the basics are easy to pick up. The most important thing to remember is that everything has to be a path in the SVG. You can do this by selecting the objects and going to Path > Object to Path. If you've used stroke effects (such as to make thick lines or rounded line ends, you'll need to go to Path > Stroke to Path. If you're including some raster images (such as JPEGs or PNGs), you'll need to use Path > Trace Bitmap to convert them into paths, but this can be error-prone – it's best to avoid raster images if possible.

EasyEDA can import SVG files directly. However, this feature is horribly limited, as it tries to simplify the SVG as it imports them, and the end result is usually a strange blobby mess. The simple workaround for this is to first convert the SVG to a JPEG, but this reduces definition and loses the scale of a drawing. You can import a JPEG at any size, but it can be hard to retain the exact scale you wanted.

Above
We tested this design out on Bristol Hackspace's CNC mill while while we waited for the final version from the fab

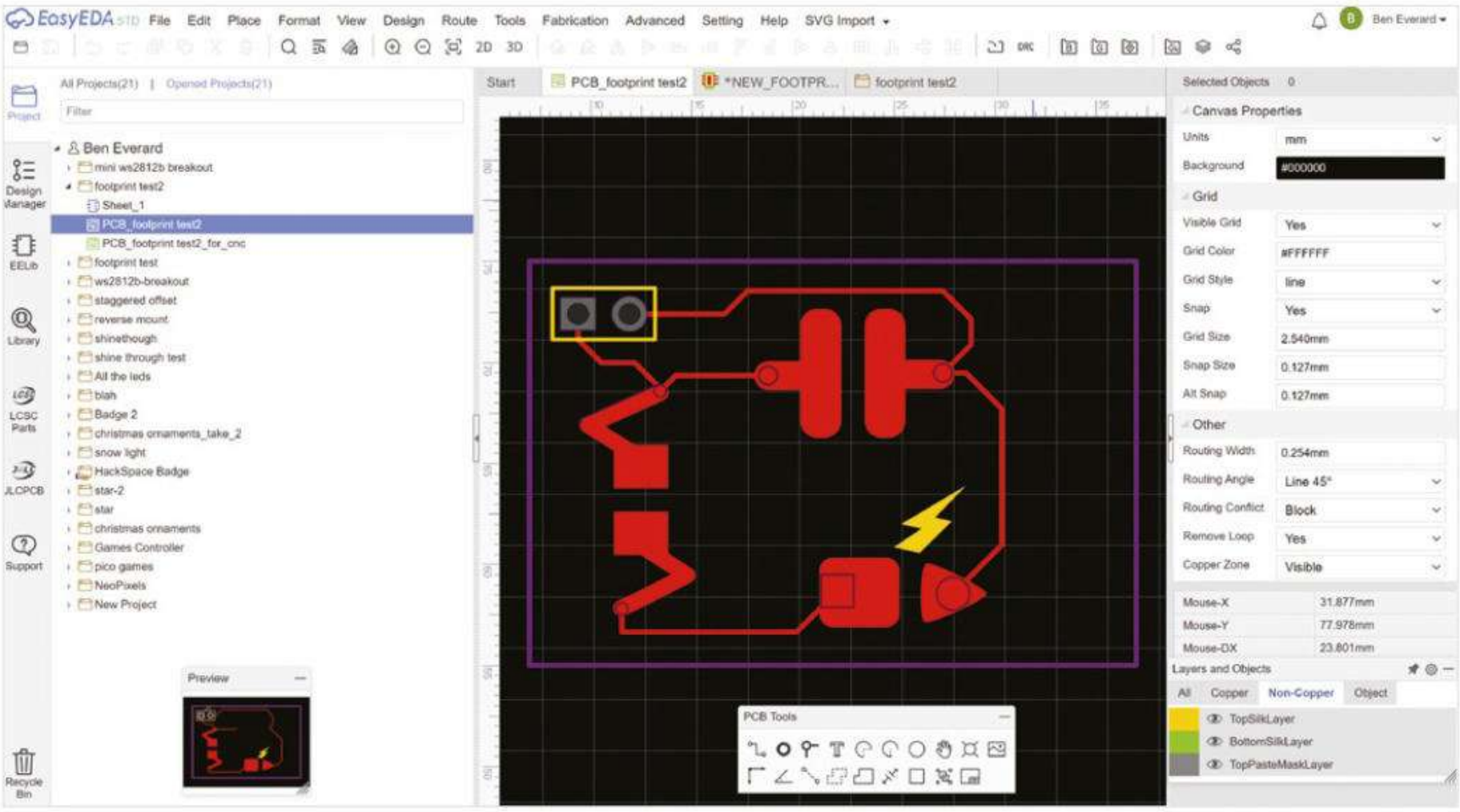
FINDING EXISTING FOOTPRINTS

A slightly strange quirk of EasyEDA is how you find footprints that you've already created. You might think there's a list of them that you can open, similar to how the projects are organised, but this isn't quite the case. Footprints are saved in the workspace. To get to them, you have to open the library window, use the search engine 'EasyEDA', the type 'Footprint', and the classes 'Workspace' – then you should be able to find the different footprints you've created.

There is the ability to edit footprints; however, we haven't found this to be particularly reliable. Sometimes, the edited version hasn't been picked up by the PCB editor even after it's been saved. We're not sure if there's a caching issue, or if there's some process we're not aware of to force the updates through, but we'd recommend including a version number in the footprint name to ensure that you're really using the one you want to.

Beautiful PCBs with custom footprints

TUTORIAL



Above
Our final PCB design, ready to be made

Fortunately, there's a better option. The EasyEDA SVG Import Extension (**hsmag.cc/EasyEDASVG**) lets you import SVGs without this problem.

You can download the release ZIP from the above website, extract this, then in EasyEDA go to Advanced > Extensions > Extension Settings > Load Extension > Select Files and select all the files in the ZIP you've just extracted. Once this is done, you should have an SVG Import option in the top menu bar.

> Remember that you can add some things in EasyEDA as well, **so you don't have to design everything in Inkscape**

We're now set up to make the footprint in Inkscape. While we want the footprint to look good, the most important part is that it functions, so you'll need to take a look at the data sheet of the part, get out your callipers, or take a look at an existing footprint that you know works. The three we're creating are all two-pad components, which makes things a bit easier. Basically, we need two pads, and the only critical dimension is the distance between them. You can scale things when you import them into EasyEDA, but we found it easiest to get the sizes right in Inkscape.

Components can have parts on many layers, as they may need things going on the top or bottom copper layer, include silkscreens, or have holes. While it is possible to load these all in from one SVG file, it's a bit tricky to manage this once it all comes into EasyEDA. You could have these in separate files, or have them all in one file, but save SVGs of all the different layers so that you can load them one at a time. We've made things easy for ourselves by only having one layer. Remember that you can add some things in EasyEDA as well, so you don't have to design everything in Inkscape if you don't want to.

CREATING A NEW FOOTPRINT

In EasyEDA (as with most EDA tools), footprints are separate from schematic components. You design a schematic and, by default, EasyEDA will assign a footprint to that so that when you create the PCB, all the components will be added. However, you can change the footprint associated with the component before you add it to the PCB. We don't

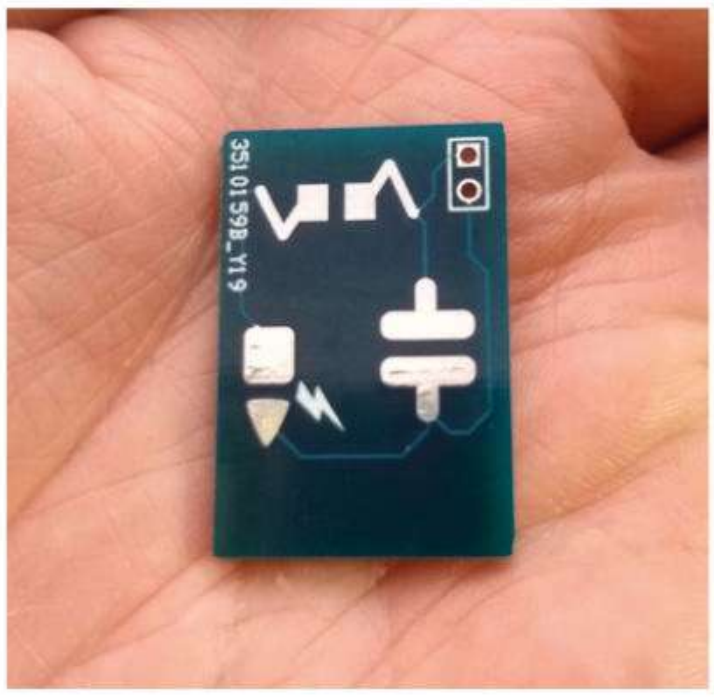
need to worry about the schematic symbols for our components, as we'll link them to already existing components. We just need to design how it will look on the PCB.

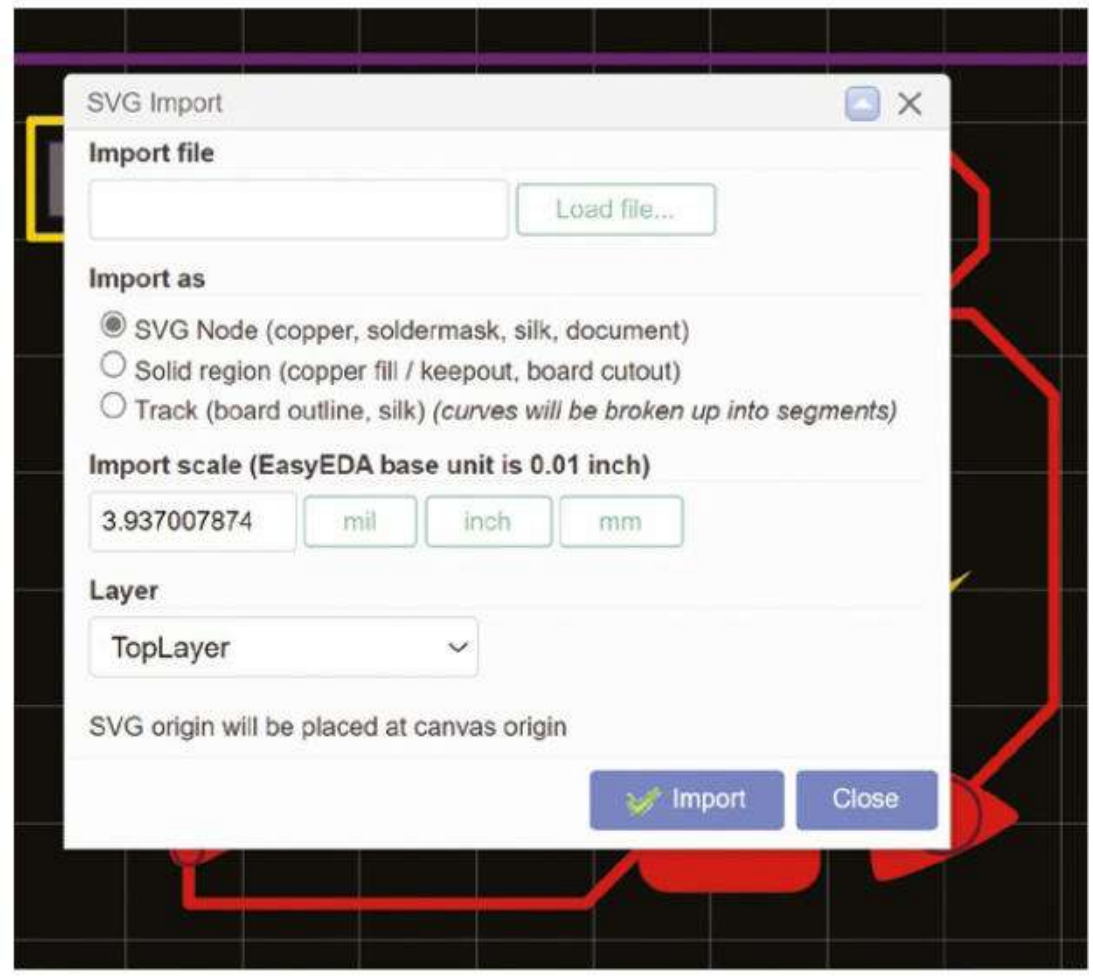
Start by creating a new footprint with File > New > Footprint. You can now use the SVG Import box to bring in the different layers that you created in Inkscape one file at a time. As you bring them in, you might want to move them on the canvas to be at, or near, the origin. Each SVG object is imported separately, so you can move them around if you like.

Remember that if you want exposed copper (or gold / solder contacts, depending on your PCB manufacturing technique), you'll need to add the shapes to both the copper layer and the solder mask layer. Since the copper layer is additive and the solder mask layer is subtractive, having the same shape in both the copper layer and the solder mask layer results in the part being exposed copper. We create the shape we want in the copper layer, then just copy-paste all the items into the solder mask layer.

Once you've got all the layers set up as you'd like, you need to create some pads. These are the bits that EasyEDA will connect to. Without these, it doesn't know where the traces should join the footprint.

Select the pads tool in the toolbox, and place one. By default, it'll be a through-hole pad, which probably isn't what you want. Once it's placed, press **ESC** to leave the pad tool and then select the pad. In the Pad Properties box, you can now change it from multilayer to top layer (or bottom if you'd prefer), and then set the size and shape as necessary. We include the pad shape inside one of our drawings that's imported into the top layer. In two-pad components, you should make sure that the pads are numbered 1 and 2. EasyEDA can get a bit confused if you create some pads then delete them and create more, but you can change the numbers in the Pad Properties box if necessary.





Above
The defaults are fine for most purposes. Just make sure you select the correct layer

Left
The final version ready for soldering

ADDING TO THE SCHEMATIC

Create a schematic as you normally would using whatever components are appropriate. Once you're ready to create the PCB, it's time to switch the footprints over to our newly created ones. Select the component and in the Custom Properties box on the right-hand side you'll see the Footprint option. Click in here and it'll open the Footprint window. You can search for your footprint name in the Class 'Workspace' to find the one you've just created. You also need to ensure that the pads correctly correspond to the ones on the schematic, but you can alter these if necessary.

Once you've set all your components up with the right footprints, you can go ahead and create a PCB as usual. You might get some DRC warnings about having tracks too close to images, but you can safely ignore these (we haven't worked out a way to stop them appearing – if you know, please let us know).

You now have your very own set of artistically designed PCB footprints. They'll be available in your workspace for whatever projects you work on.

HackSpace
TECHNOLOGY IN YOUR HANDS

FIELD TEST

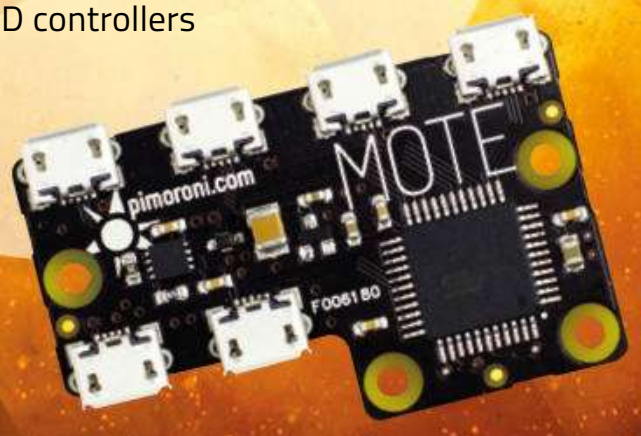
HACK | MAKE | BUILD | CREATE

Hacker gear poked, prodded, taken apart, and investigated

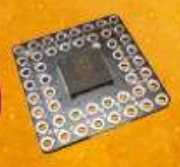
PG 102

BEST OF BREED

LED controllers



PG 108



PGA2040

Just the bare essentials to get an RP2040 running

PG 112

QT PY ESP32-S2

WiFi networking goes tiny

ONLY THE BEST

Addressable LED light strips

Easily add lots of lights to your next project

By **Marc de Vinck** @devinck

LEDs, we all love them, right? It's almost hard to imagine a project that doesn't include at least one LED as an indicator of power, status of a sensor, or even just for aesthetic purposes. But what happens when you want a lot of LEDs? Not just one, two, or even ten. I'm talking about 50 or more. It can get complicated controlling all of them, and that's where this Best of Breed comes into play. I'll be looking at addressable LED strips and accessories that allow you to control an almost unlimited number of LEDs.

And not just LED strips that you find at big box outlet stores. Those are fun, and some even have RGB colour control, but it's limited to preprogrammed sequences. Many are one colour, generally some type of white light, but there are RGB versions that allow you to run interesting patterns or change to specific colours, but you can't control each LED. I'll be looking at LED strips that allow you to individually control each LED 'pixel' with a microcontroller or computer. It greatly extends the usefulness of these types of LED strips.

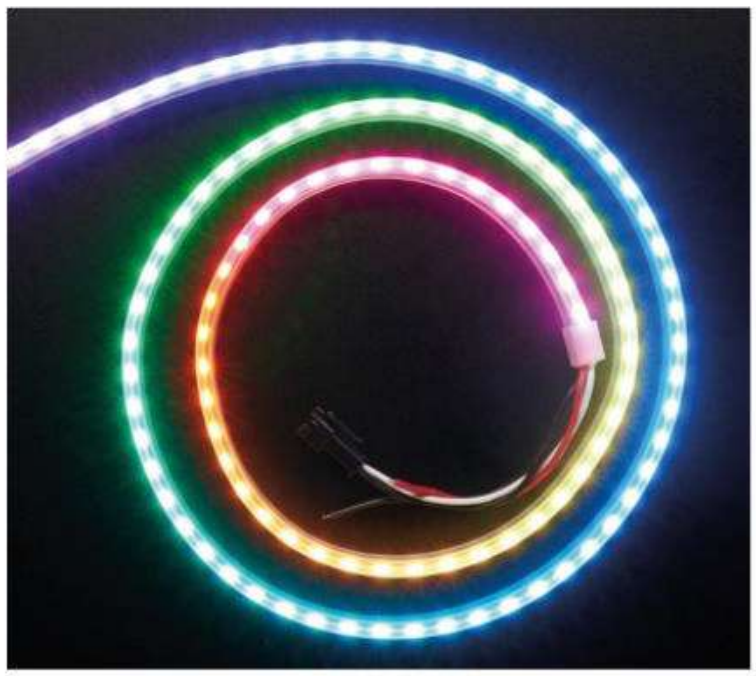
And I'll also be focusing on some of the more unique addressable LED strips. There are so many 'standard' LED strips that are controllable, just visit any online retailer of DIY electronics and you'll find a variety of them. But, recently, I've been seeing some more unusual LED strips. Versions that have incredible LED densities, are much smaller than the standard variety, or light from the side instead of the top. So, let's dive in and look at some of my favourite LED light strips and accessories.

Adafruit NeoPixel LED Side Light Strip vs Mote

ADAFRUIT $34.95 | adafruit.com **PIMORONI** £42.60 | pimoroni.com

As opposed to the standard forward-facing LEDs in most strips, the Adafruit NeoPixel LED Side Light Strip allows you to make tight turns with ease due to the unique sideways-facing LEDs. Each strip features 120 LEDs per metre mounted to a flexible black PCB material, along with weatherproof sheathing. Strips can be easily cut shorter at specific points, or you can connect each one-metre segment together to form more complex shapes via the two-pin JST SM connector on the end.

Just remember that these LED light strips have a high-density LED count, so you are going to need a good power supply. Head over to the Adafruit website for more information, example code, and a handy power usage calculator.



Left
The bendiest LED strip

Below
LEDs from any computer

The Mote, by Pimoroni, is an interesting USB-based LED light strip controller with four channels that connects to your computer. It allows you to control up to four different Mote sticks, included in this kit. Each Mote stick features 16 LEDs for a total of 64 pixels per Mote controller.

It's a simple and very quick way to get up and running with lots of LEDs via your computer. Pimoroni has a nice Mote Python library to download with examples of animations, and other resources that allow you to control the Mote via your iPhone, Android, Siri, or Raspberry Pi. And speaking of Raspberry Pi, Pimoroni also makes a pHAT version of the Mote that lets you control your Mote sticks directly through your Raspberry Pi's GPIO.

VERDICT

Adafruit NeoPixel LED Side Light Strip

Different, in a good way!

10/10

Mote

Makes adding LED strips simple.

9/10

BEST OF BREED

Mini Skinny NeoPixel Digital RGB LED Strip

ADAFRUIT $64.95 | adafruit.com

If space is at a premium with your project, you might want to look at the Mini Skinny NeoPixel Digital RGB LED Strip from Adafruit. It's their 'skinny' version of the classic NeoPixel strip. They feature 144 RGB addressable mini-LEDs per metre, and the strip is only 7.5mm wide. With that many LEDs packing into such a small flexible PCB, be sure to read more about the power requirements. If you've got the power, but not the space, check out this LED strip, it won't disappoint!

VERDICT

Mini Skinny NeoPixel Digital RGB LED Strip

Bright and tiny!

9/10

Plasma 2040

PIMORONI $14.41 | pimoroni.com

If you have some LEDs strips already, or you are planning your next project and haven't decided on a controller yet, check out the Plasma 2040 from Pimoroni. The Plasma 2040 is based around the new RP2040 IC. This board allows you to easily control a variety of addressable LED strips in a familiar development environment like C, C++, or MicroPython.

The board is programmable via a standard USB-C connection, and since it's USB-C, it can also power your LED strips with up to 3A of available power. It also features three user-assignable buttons, a QW/ST connector for plug-and-play with Qwiic or STEMMA QT boards. All these features make the Plasma 2040 an economical and sound choice for your next project.

Above
Space-saving blinkies

Left
Add the power of RP2040 to your LEDs

VERDICT

Plasma 2040

Simple and affordable.

9/10

Above
A programable LED necklace

EightByEight Blinky

BLINKINLABS $36.90 | tindie.com

This might not be an LED strip, but while searching for LED strips, I came across the EightByEight Blinky from Blinkinlabs and just had to include it in the roundup. The board features 64 independently controllable RGB LEDs, an ESP8266, an accelerometer, and a rechargeable lithium-ion battery. And all of that is packaged beautifully in this well-designed solid black PCB with gold-plated accents. I love the look of this PCB!

So, what can you do with the EightByEight Blinky? A lot! This is because it can be programmed just like an Arduino, and each LED can be any colour of the rainbow, all while reacting with the accelerometer. It certainly looks like a great board for an interactive wearable project! The EightByEight Blinky comes fully assembled, just add a battery and get to programming!

VERDICT

EightByEight

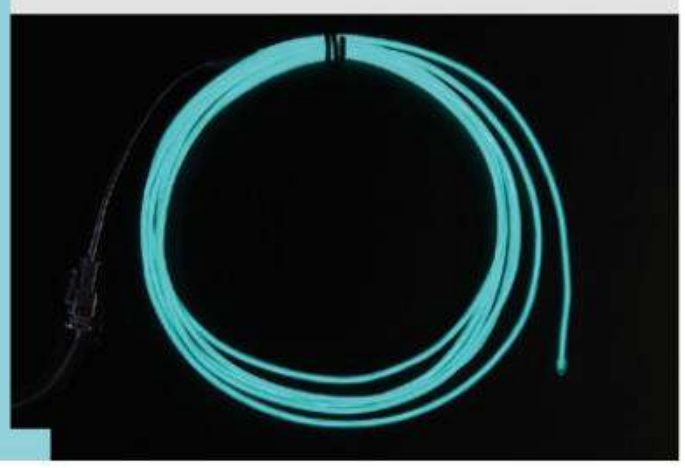
Perhaps the most stylish LED matrix available

9/10

EL WIRE STARTER PACK

ADAFRUIT $19.95 | adafruit.com

I know what you are thinking: 'This is not an LED!' And you're right. But I thought I would include the EL wire starter pack in this roundup since it's basically a cousin, OK, maybe second cousin, to the LED strip. EL wires are typically a lot thinner in diameter, a bit more fragile, but also a bit more unusual looking. If you've never used one, it's worth picking up a strip and power supply. They are a lot of fun!



BEST OF BREED

12mm Diffused Flat Digital RGB LED Pixels

ADAFRUIT $39.95 | adafruit.com

When you think of LED light strips, you typically think of surface-mount LEDs on a long flexible PCB-like backer. But that's just one kind of LED strip. The 12mm Diffused Flat Digital RGB LED Pixels from Adafruit are a bit different because they lack the flexible PCB backer. Instead, these light strips use flexible wires that span larger distances between each addressable RGB LED. This means you can typically cover a larger area of light using less power per metre. You can also navigate these LEDs inside projects where typical LED strips wouldn't be able to bend. Check out the Adafruit website for more information about the power requirements, example code, and more.

Left
Like fairy lights, but programmable

VERDICT

12mm Diffused Flat Digital RGB LED Pixels

Sometimes fewer LEDs are better!

9/10

NEOPIXEL SIDE-LIGHT

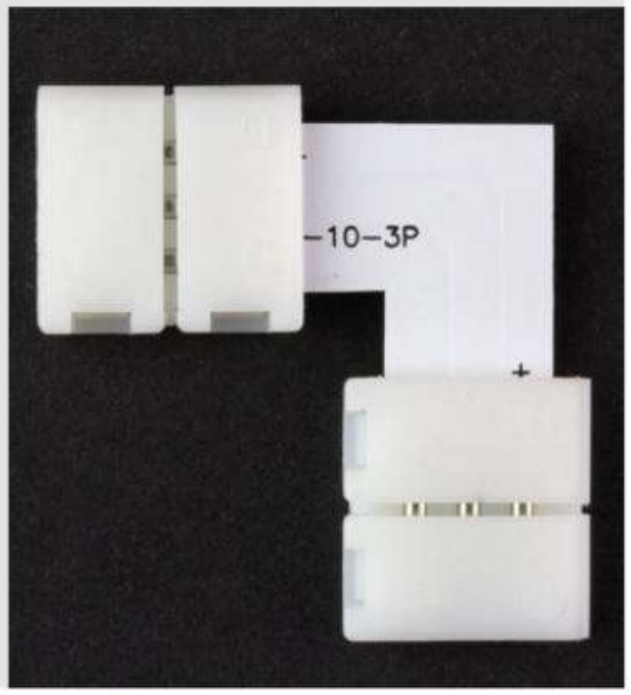
ADAFRUIT $3.95 | adafruit.com

Sometimes you just want to roll your own LED strip, or have a project that was inspired by one of those interesting side-lit LED strips. Don't worry, Adafruit sells loose side-lit NeoPixels with integrated drivers. These are great for small indicators in tight spaces since it's only 4mm × 2mm. Grab a ten-pack and get to making your own custom RGB LED strip!

FLEXIBLE LED STRIP ADAPTOR

PIMORONI $1.34 | pimoroni.com

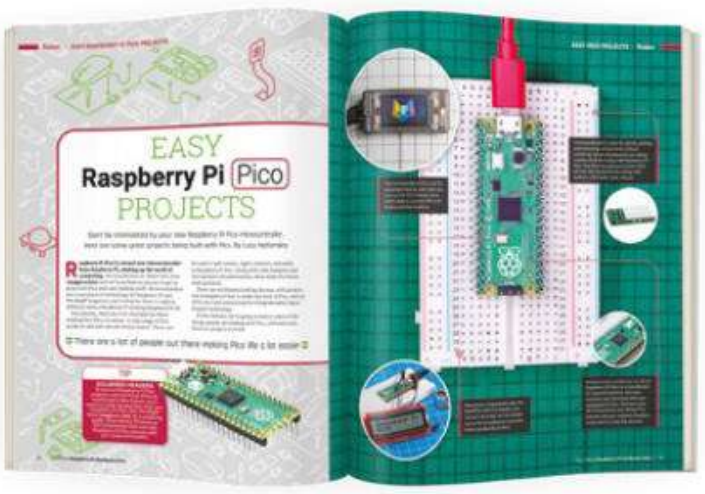
Here is a handy little device that allows you to connect many of the three-pin standard LED strips together without the need to solder. They have straight, cross, and a 90° turn versions of these connectors. Typically, you need to solder small wires to the LED strips to make complex bends and connections, but not anymore! Just place the end of the LED strip in the connector and snap it closed. Just be aware that these only work with certain types of LED strips. Head over to the Pimoroni website for more info.



THE *Official* RASPBERRY PI HANDBOOK 2022

200 PAGES OF RASPBERRY PI

- QuickStart guide to setting up your Raspberry Pi computer
- Updated with Raspberry Pi Pico and all the latest kit
- The very best projects built by your Raspberry Pi community
- Discover incredible kit and tutorials for your projects



Buy online: magpi.cc/store

REVIEW

PGA2040

Just the bare essentials to get RP2040 working

PIMORONI £6.90 | shop.pimoroni.com

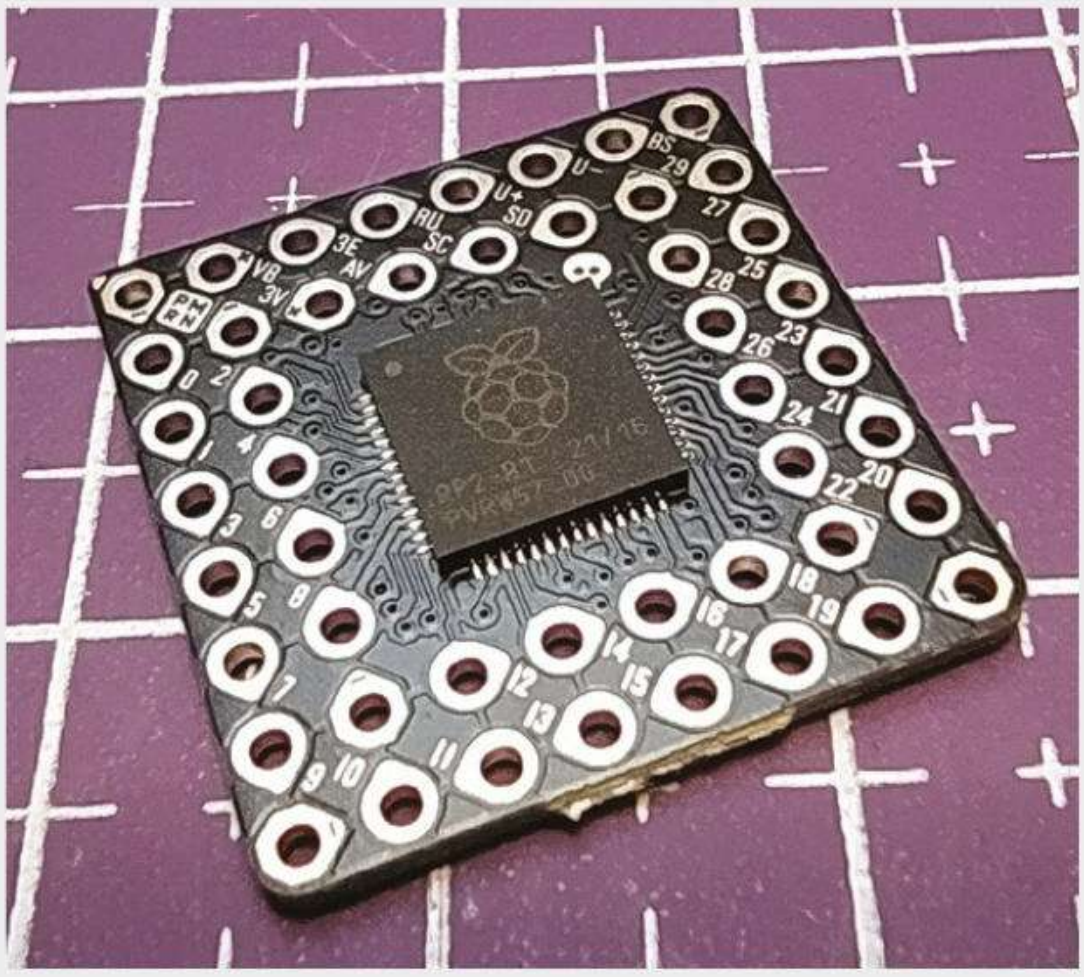
By **Jo Hinchliffe** @concreted0g

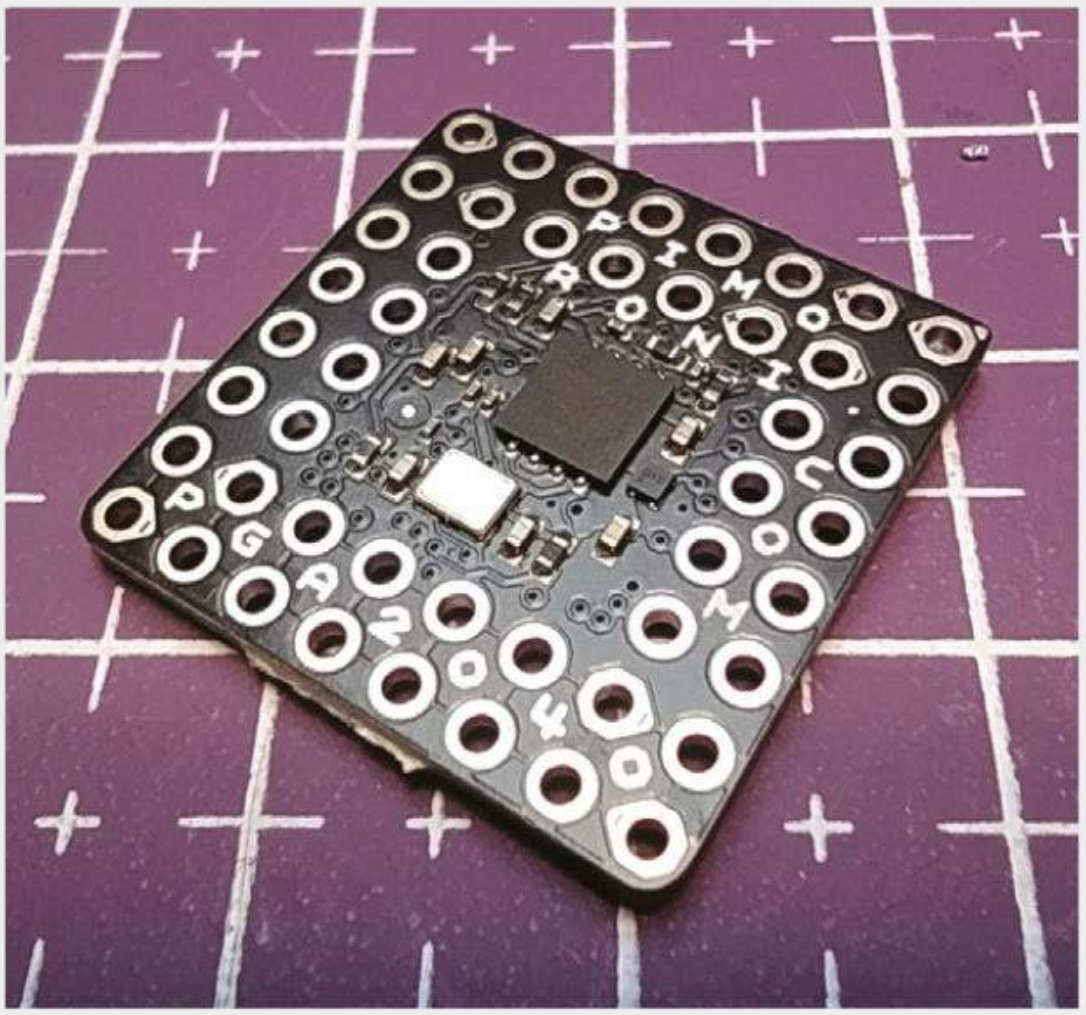
The PGA2040 from Pimoroni is a tiny RP2040 breakout board squashed down to just 21 mm square. In terms of functionality, it's pretty much identical to a Pico but, actually, despite its stamp-like status, it has more I/O pins broken out for use! In case you haven't come across the RP2040 specification, in a nutshell, this PGA2040 features a dual-core Arm Cortex-M0+ which can run up to a speedy 133MHz. There's 265kB of SRAM on board, I2C, SPI, and UART, plus USB support – more on that later. There are 30 GPIO pins featuring PIO that can be used to create custom hardware interfaces, which are proving a popular approach for some amazing projects in the Pico community.

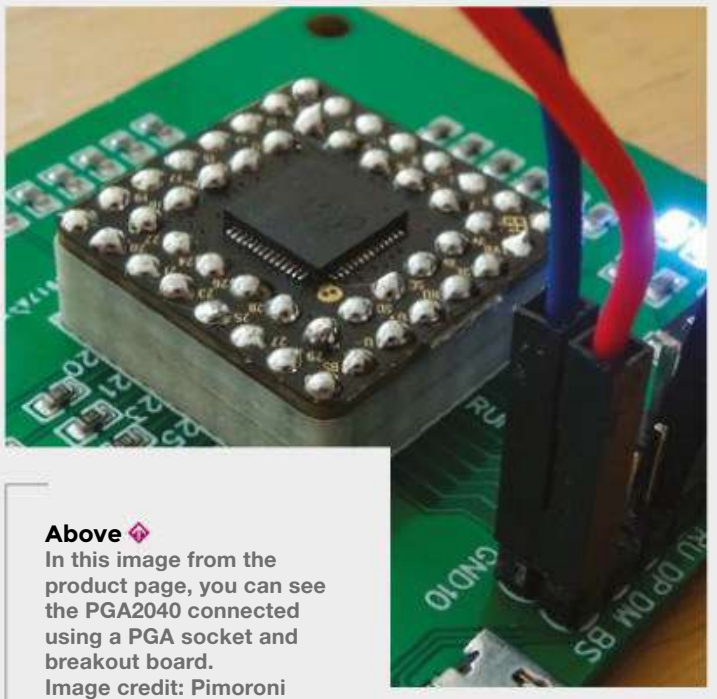
The PGA in PGA2040 stands for Pin Grid Array and was a pretty common integrated circuit package standard with lots of the big chip companies a few decades ago. It's a robust standard in that it uses common 2.54 mm spacing (therefore suitable for common header pins) and can pack a lot of pins into a small area. It's distinctly retro-looking and, with its PGA layout, the PGA2040 should catch the attention of those wanting to make vintage-looking boards and projects. We'd love to see a vintage operating system emulated with this board. It's not just nostalgia, however: the PGA layout makes a lot of sense as it's a small and mechanically strong board for your project ideas. Of course, you can also find all kinds of PGA standard prototyping boards, and all manner of PGA sockets out there to incorporate into your ideas. Whilst we are on aesthetics, although this board is tiny, close inspection reveals some beautiful silkscreen design and a very pretty approach to pin labelling.

Just like the Raspberry Pi Pico, the PGA2040 is firmware agnostic and can be happily programmed in MicroPython, C, or CircuitPython and, as such, there is a wealth of support and documentation out in the wild. Anything you've already created on a breadboard with a Pico is likely to work perfectly, as are established projects written by others with the PGA2040, albeit with its different hardware footprint. The observant will notice there's no on-board USB port! To connect a USB, you only need to connect to four pins, plus one more momentarily. Brilliantly, all the pins for the USB are all along one side of the PGA2040, so you have the choice of adding just one row of headers rather than fully populating the entire board with header pins should you not want to. You may also want to create a project that doesn't require a USB socket permanently connected, so we wanted

Below
The top side of the PGA2040 adorned with the RP2040 chip







Above
In this image from the product page, you can see the PGA2040 connected using a PGA socket and breakout board.
Image credit: Pimoroni

Left
The reverse side of the board, with all the components that make it Pico-compatible

VERDICT

An excellent Pico-compatible product that is easy to use and makes tiny projects possible.

9/10

to explore a non-header-pin approach to working with the board. For a hardware USB socket, we picked up a Pololu Micro-B Breakout Board, also stocked by Pimoroni, and soldered the pin headers to it to make it breadboard-compatible. We were then able to use our breadboard DuPont cables, together with some miniature claw-type test hooks, to connect up to the PGA2040.

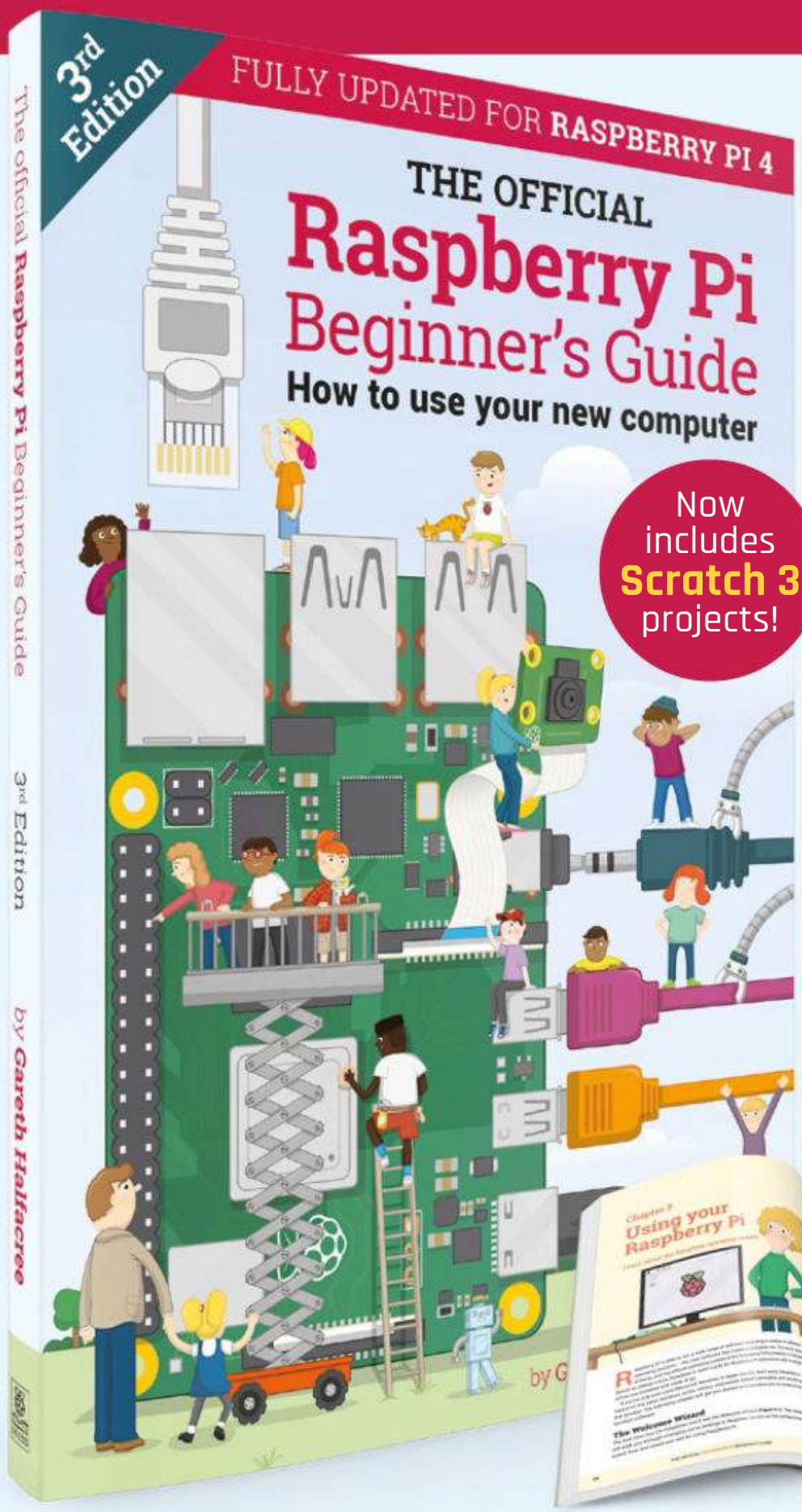
Wiring it up is pretty straightforward, connecting the V-Bus, GND, and Data lines + and – on the PGA2040 to the same pins on the USB connector. As our PGA2040 had never been used before, we needed to connect it to a computer in BOOTSEL mode. On a standard Pico this is achieved by holding down the BOOTSEL button whilst connecting the Pico via USB. The BOOTSEL button on a Pico just connects the BOOTSEL pin to ground and, as such, you can emulate this by simply wiring the BOOTSEL pin on the PGA2040 to connect to ground on the USB connector whilst you plug in the device. Once it appears on your computer as a drive, you can disconnect that pin. When connected, opening the Thonny editor on our machine saw the PGA2040 recognised as already attached. The usual prompt to install the MicroPython firmware flashed up on screen and the ensuing installation process all ran perfectly. Within seconds we were up and running, with the REPL awaiting our commands.

If you are new to Pico, you might want to look through the many tutorials and examples we have had in previous issues of the magazine or, alternatively, you could work through the examples in the Raspberry Pi book *Get Started with MicroPython on Raspberry Pi Pico,* written by Gareth Halfacree and our very own Ben Everard, (**hsmag.cc/GetStartedMicroPython**).

Once you are up and running with some Pico knowledge, the PGA2040 is a great way for you to make your projects smaller and potentially with a vintage aesthetic.

Right
Using claw-type miniature test probes, it is straightforward to connect to the PGA2040

THE OFFICIAL
Raspberry Pi
Beginner's Guide



The only guide you need to get started with Raspberry Pi

Inside:

- Learn how to set up your Raspberry Pi, install an operating system, and start using it
- Follow step-by-step guides to code your own animations and games, using both the Scratch 3 and Python languages
- Create amazing projects by connecting electronic components to Raspberry Pi's GPIO pins

Plus much, much more!

£10 with FREE worldwide delivery

Buy online: magpi.cc/BGbook

REVIEW

QT Py ESP32 S2

Squeezing a lot of performance into a tiny form factor

ADAFRUIT from $9.95 | adafruit.com

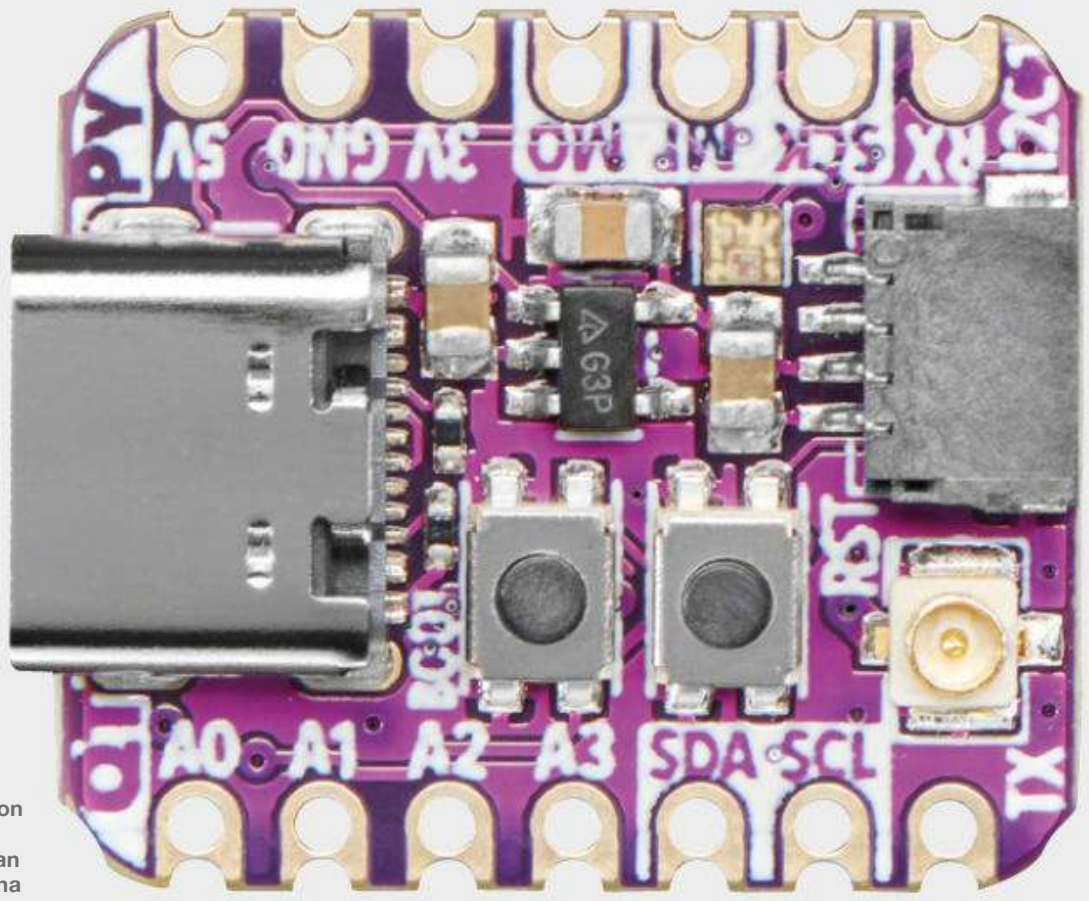
By **Ben Everard** @ben_everard

ESP-based microcontrollers from Espressif have, more or less, defined the DIY IoT scene for longer than HackSpace magazine has been around. 'Stick an ESP on it' has been a standard phrase for adding internet connectivity to a device. While the classic ESP32 devices aren't really difficult to use, they're not as easy as some other microcontrollers that have drag-and-drop programming with CircuitPython. This changed a couple of years ago with the introduction of the ESP32-S2 which lacked some features of the original, but added native USB support that allowed easy programming.

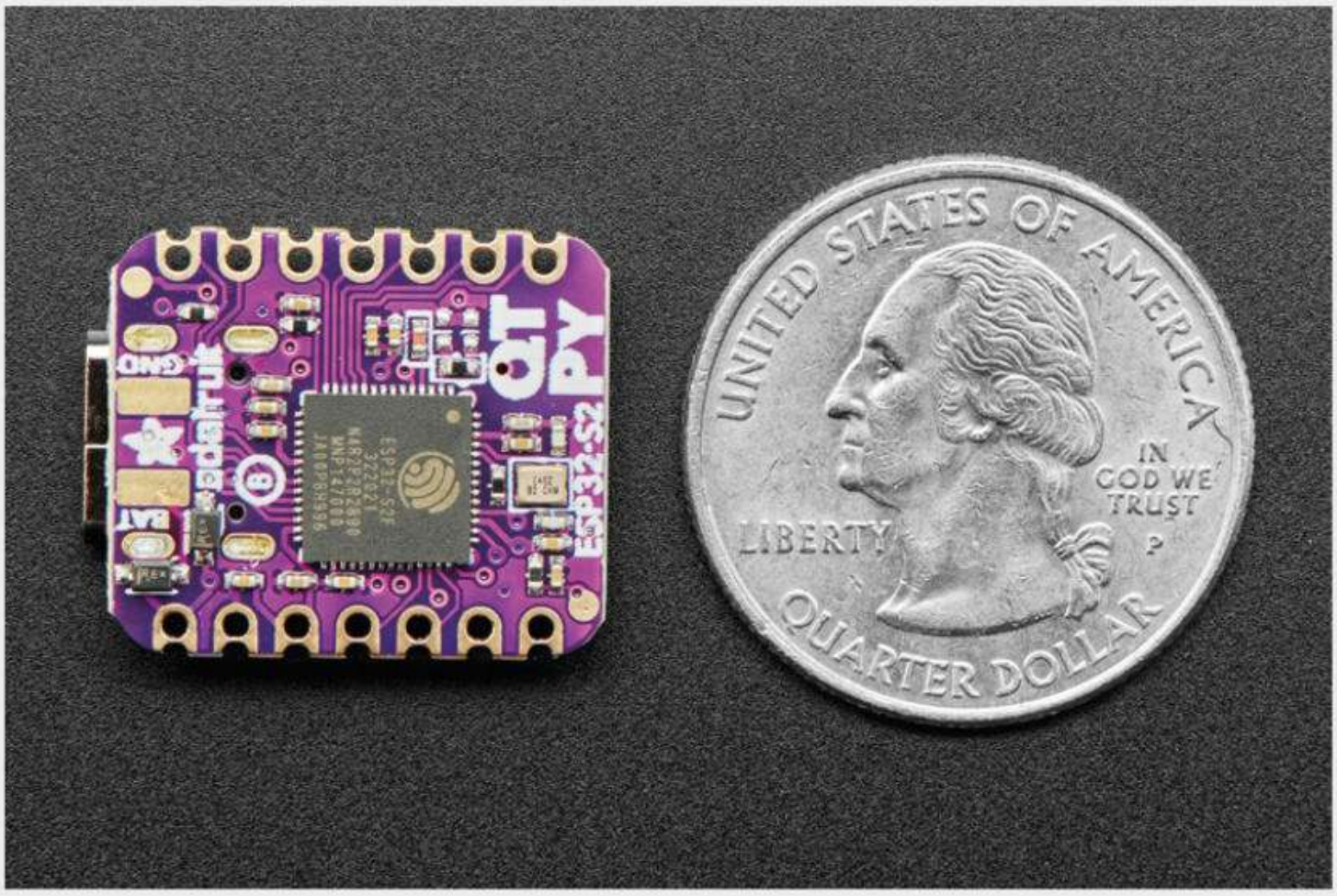
Essentially, this means that in ESP32-class devices, at the moment, there's a trade-off. The original is more powerful and has both WiFi and Bluetooth, but is slightly more complex to program. The S2 is less powerful and only has WiFi, but is easier to program. Given that the ESP32 is a powerful processor anyway, for most purposes, we haven't been limited by the processing power. Bluetooth is another of those things that is completely unused most of the time, but is occasionally really useful.

The QT Py series of boards from Adafruit bring popular processors into a very small form factor (it's compatible with Seeed Studio's XIAO). This time, it's the turn of the ESP32-S2 to get the QT Py treatment. Despite its teeny size, there are eleven GPIOs (plus an additional two on the STEMMA QT connector). Ten of these are 12-bit analogue inputs, and one is an 8-bit analogue output. There's hardware I2C, SPI, and UART, and capacitive touch. All in all, there's a lot of connectivity for such a small device, and we haven't even got to WiFi yet.

Jamming so much onto such a small device does mean that there are some trade-offs. There are no mounting holes, so attaching it to your project means either soldering down or a blob of hot glue. Although the pads are castellated, there are components on the bottom of the PCB, so it's not easy to surface-mount it. The power connections (other than USB) are pads on the bottom of the PCB.



Right
There's a version with a U.FL connector for an external antenna



Left
The microcontroller is mounted on the underside of the PCB

This means you have to 'surface-mount' any wires if you want an additional power connection. It's not a particularly difficult solder connection, but it's a little more tricky than soldering wires through-hole. For many projects, none of these will be particularly limiting, but it's worth being aware of them.

In use, the QT Py behaves much like any other ESP32-S2 board. You can drag and drop code and libraries onto a USB device created when you plug it in, and it's compatible with a huge range of libraries. You can also program this with the Arduino IDE if you prefer. The two things that really make the QT Py stand out against other ESP32-S2 boards are size and price. At $12.95 (or $9.95 if you want the option with an external aerial), it's one of the cheapest ways of getting this particular microcontroller.

> "There are a whole host of non-technical features that often define success – **documentation, ease of use, price**"

WHIPPERSNAPPER

The native USB feature of the ESP32-S2 makes it much easier for beginners to get started. If you're a veteran microcontroller user, you may now barely think about the differences between native USB and using a serial-to-USB setup. However, when getting started for the first time, it's the difference between just needing to install a text editor and needing to install drivers, an IDE (or some uploading software) working out which serial port is in use (maybe working out what serial ports are in the first place), and a myriad of debugging steps that can put new users off.

We often think of a microcontroller's functionality in terms of technical features, but there are a whole host of non-technical features that often define success – documentation, ease of use, price. After all, if you can't get the thing working, it's irrelevant whether or not it's technically possible for it to perform its task. The exact importance of the different levels of these 'soft' features varies hugely from one maker to another. The QT Py ESP32-S2 has brought down the lower bar for cost (it's about half the cost of other CircuitPython-compatible ESP32-S2 boards), while retaining ease of use, enabling a whole new segment of makers to 'stick an ESP on it'. ◻

VERDICT

All the power and ease of use we've come to expect in ESP32-S2 boards made tiny and cheap.

10/10

next month

issue
#53
ON SALE
24 MARCH

UPCYCLING

ALSO

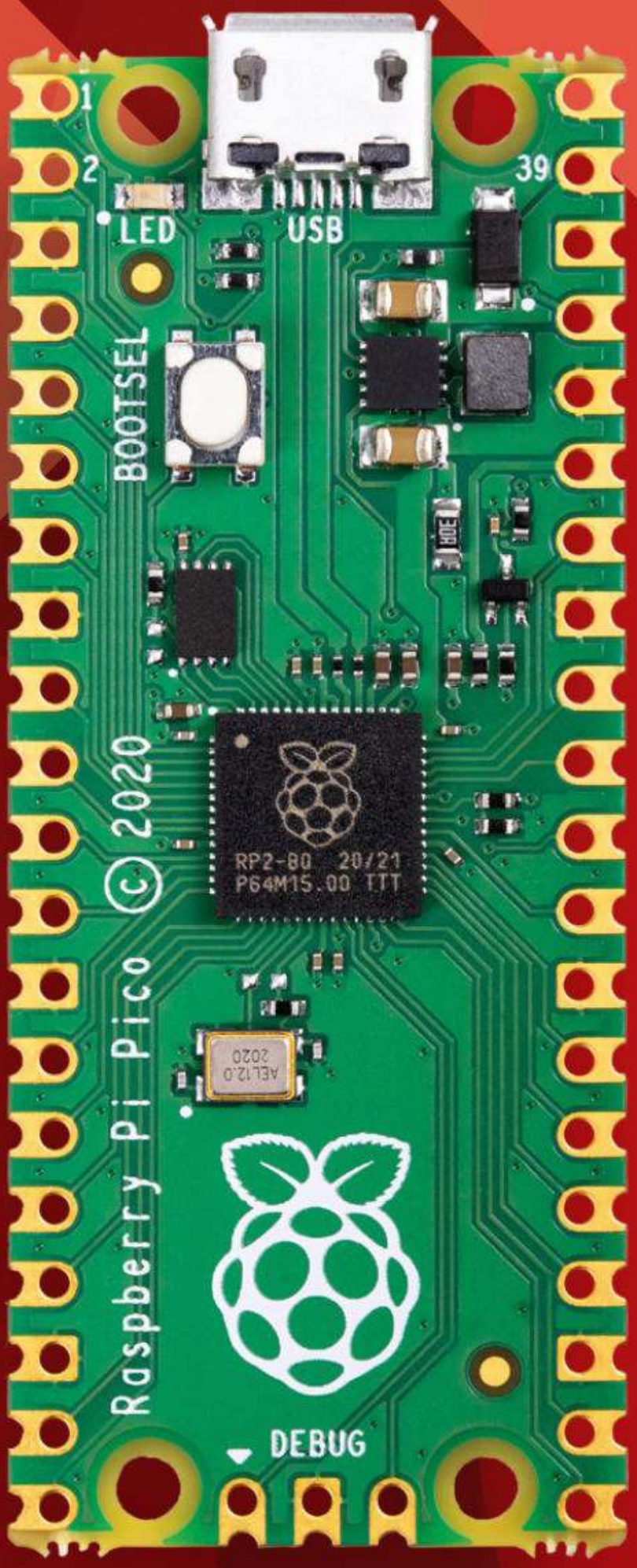
- 3D PRINTING
- RASPBERRY PI
- LASERS
- SOLDERING
- AND MUCH MORE

DON'T MISS OUT

hsmag.cc/subscribe

"The best way to learn is by failure. When something fails, your mind will start thinking of 100 reasons why it’s failing, to find the source of the problem. You end up testing all sorts of aspects of the design, which, if it didn’t fail, you wouldn’t even think about. Failure is an important part of the process."

Carl Bugeja


1
2
39
LED
USB
BOOTSEL
RP2-B0 20/21
P64M15.00 TTT
AEL12.0
2020
Raspberry Pi Pico © 2020
DEBUG